Le Campus

WordPress 3

Toutes les clés pour créer, maintenir et faire évoluer votre site web



Xavier Borderie
Francis Chouquet
Amaury Balmer

Préambule de Michel Valdrighi
Préface de Matt Mullenweg

PEARSON

WordPress 3

Toutes les clés pour créer, maintenir et faire évoluer votre site web

PEARSON

Publié par Pearson Education France
47 bis, rue des Vinaigriers
75010 PARIS
Tél. : 01 72 74 90 00
www.pearson.fr

Collaboration éditoriale : Hervé Guyader

Mise en pages : TyPAO

ISBN : 978-2-7440-4162-4

WordPress 3

Toutes les clés pour créer, maintenir et faire évoluer votre site web

Xavier Borderie, Francis Chouquet et Amaury Balmer

PEARSON



Table des matières

Partie I

Partie III

WordPress côté développeur : concevoir une extension

Partie V
Les autres projets de la WordPress Foundation 433

Partie VI

Préambule

L'odeur du maquis corse, le ballet des hirondelles à ma fenêtre, une conversation avec une blogueuse australienne ; treize heures un samedi, et la décision de créer mon propre outil de blog était prise.

C'est un désir d'indépendance qui est à l'origine de b2 : j'utilisais Blogger.com et un service externe pour les commentaires, dont le nom m'échappe – comme m'ont échappé les nombreux commentaires qui y étaient hébergés quand le service a fermé.

Mes buts : conserver la souplesse d'utilisation de Blogger, sans les indisponibilités temporaires du serveur, et m'éclater. Les Anglais de Blur chantaient "Song 2" dans Winamp, je criais "Woohoo", et "Blog 2" est né, renommé "b2" dans la soirée.

En juin 2001, il existait déjà un certain nombre d'outils de blog à installer soi-même, mais très peu étaient vraiment open-source, et ceux qui l'étaient ne me convenaient pas : gabarits difficiles à éditer, interfaces trop basiques ou trop encombrantes, installations souvent malaisées.

Aussi, quand j'ai mis en ligne un premier prototype utilisable, j'ai immédiatement sollicité l'avis et les attentes d'autres blogueurs pour déterminer ce qui serait et ne serait pas présent dans b2.

De cette première période il reste un souvenir d'effervescence et d'excitation face à l'inconnu, exacerbé au fil des semaines par la première version publique, le premier Blogathon (24 heures de blog d'affilée, pour une bonne cause), les premières contributions d'autres développeurs, autant d'étapes qui avec le temps prennent l'aspect de mythes fondateurs.

Fin 2002, la dépression me gagnait et je n'ai plus donné de nouvelles que sporadiquement. J'ai pu prendre conscience de l'importance du projet (et de l'importance de l'Open Source) lorsque, contrairement à d'autres avant lui, il n'est pas mort avec ma disparition d'Internet.

Au printemps 2003, une fois la dépression vaincue, je découvris que les bourgeons de mon code avaient donné trois fruits : WordPress, b2++ et b2evolution.

La familiarité et les affinités avec les développeurs de WordPress me poussèrent à préférer celui-ci, et j'annonçai la fin de b2 et sa succession par WordPress. Je n'avais plus beaucoup de temps pour diriger un projet, aussi j'étais heureux de contribuer au développement par mes idées et mon code au fil des mois suivants, sous la direction de Matt Mullenweg et Mike Little.

Ironiquement, pour un projet aux racines françaises, WordPress ne disposait pas d'une traduction française (la blogosphère francophone était à peine naissante en 2001). Un oubli rectifié en 2004, notamment par le travail de Xavier Borderie et par la formation de l'équipe WordPress-FR.

Quand je regarde en arrière, après toutes ces années et maintenant que WordPress est assez célèbre (assez pour que vous teniez ce livre entre vos mains !), je ne peux qu'être fier du chemin parcouru.

Il est une idée reçue qui ferait d'Internet et des blogs un monde virtuel. C'est oublier qu'il s'agit d'un monde peuplé par des personnalités réelles ; les relations qui se créent entre elles sont tout sauf virtuelles.

Je n'aurais jamais assez de pages pour vous décrire à quel point b2, WordPress et les rencontres que j'ai pu faire grâce à ces projets ont changé ma vie.

Il ne me reste qu'à souhaiter que WordPress change autant la vôtre !

Michel Valdrighi
Créateur du projet b2 à l'origine de WordPress
Octobre 2008

Préface

Ma participation au mouvement open-source et au projet qui deviendra par la suite WordPress a commencé de manière assez humble. Le Français Michel Valdrighi avait créé l'excellent outil b2/cafelog, que j'utilisais pour mon propre blog, et dont j'ai commencé à modifier le code source pour mes besoins. Autant que je me souvienne, ma première contribution fut un petit bout de code que j'avais baptisé Texturize et qui tentait autant que faire se peut de transformer les signes de ponctuation basiques en entités typographiques adéquates. L'idée était de faire gagner du temps aux utilisateurs de b2, en faisant en sorte qu'ils n'aient pas à se soucier de ces entités typographiques, ou même à se douter de leur existence, le tout se faisant de manière automatique. Quand ce bout de code a été accepté, j'ai ressenti une vague d'émotion à l'idée que des centaines de sites web allaient utiliser le code que j'avais écrit. L'euphorie devint une drogue – j'étais devenu accro à l'Open Source.

Du concept même de l'Open Source, qui stipule que le code source d'un programme appartient à qui veut bien s'en occuper, découle une idée magnifique : un projet n'est plus lié à un seul auteur, et il vivra tant que des développeurs en auront l'envie. C'est ce qui est arrivé à b2/cafelog, abandonné par Michel et que j'ai repris en main avec l'aide de Mike Little afin d'officialiser toutes les améliorations que nous voulions y apporter. La tâche était énorme, mais par chance nombreux furent ceux qui y crurent et se mirent à participer : en nous attachant à quelques idées fondamentales, nous avons rapidement fait renaître autour de WordPress la communauté originale de b2.

La première de ces idées était de respecter le temps des utilisateurs. WordPress est une page blanche qui ne sert à rien tant que vous n'y apportez pas votre créativité et votre temps. Un logiciel doit être un moyen de parvenir à une fin et non être une fin en soi. Cela implique que, si nous faisons bien notre travail en tant que développeurs, ce travail devient invisible, et il n'y a plus aucun obstacle entre l'auteur et ce qu'il écrit.

Deuxièmement, nous avons décidé de miser totalement sur les standards du Web, qui à l'époque commençaient tout juste à se faire remarquer. En encourageant l'utilisation de standards comme le XHTML, le CSS, le RSS 2.0, ainsi que les microformats comme le XFN, nous voulions montrer le bon exemple pour les autres projets existants, et faire du Web, ou du moins le petit coin du Web que nous occupions, un endroit où il fait bon se rendre.

En troisième lieu, nous voulions garder en tête que de nombreux projets open-source souffraient d'une asphyxie due à un trop grand nombre d'options. L'Open Source est un mouvement prônant un plus grand contrôle des outils, ce que certains projets tentaient de donner en proposant un très grand nombre d'options. Pire encore, ils se servaient souvent des options pour résoudre les débats internes ; par exemple, s'il y avait un profond désaccord dans la communauté sur la taille à donner à une section de l'interface, ils finissaient par couper court au débat en créant une option, afin de laisser l'utilisateur décider, plutôt que chercher à savoir quel choix était le meilleur. Trop d'options poussent à une surcharge cognitive avec laquelle l'utilisateur ne souhaite pas toujours se confronter, et qui au final le ralentit.

La quatrième idée tenait à notre conviction que coder est un art (en anglais, *code is poetry*, littéralement "le code est un poème"). Cette idée nous avait été inspirée par le poète T. S. Eliot, qui, avec une économie de mots sans pareille, parvenait à donner plusieurs niveaux de sens à chaque phrase et syllabe qu'il écrivait. Le code peut s'en approcher : court, élégant, ne révélant que le nécessaire, et devenant une source d'inspiration. En gardant cette idée en tête, la communauté formée autour de WordPress a réussi à conserver le cœur du code léger et véloce, même après des années à constamment étendre le nombre de ses fonctionnalités.

Enfin, il y a également une envie de mettre du beau et du respect dans notre travail, chose que nous commémorons lors de chaque version majeure en la baptisant du nom de l'un de nos jazzmen favoris.

À ce jour, WordPress fait tourner plus de 6 millions de sites web, ce qui fait 5,999 millions de plus que ce que je n'aurais jamais pu imaginer au début du projet. Des blogueurs connus et inconnus partagent leur travail avec le monde par le biais de WordPress, au rythme de plusieurs centaines de milliers d'articles par jour. L'infrastructure de création d'extensions et de thèmes, créée quasiment par hasard, est maintenant au cœur d'un écosystème en pleine croissance, fait de milliers de développeurs concevant des fonctionnalités qui transforment WordPress en tout autre chose, parfois sans ressemblance avec le logiciel d'origine. Ce logiciel est devenu la base du travail d'autres développeurs, comme une autre page blanche.

À un tel niveau de croissance, nombre de projets commenceraient à s'écrouler sous leur propre poids, à ralentir leur progression ou à s'empâter, mais la beauté de l'Open Source fait que notre communauté se rapproche plutôt d'une boule de neige descendant une pente : nous prenons de la vitesse. WordPress voit désormais naître trois versions majeures par an, chacune répondant aux demandes explicites ou implicites d'utilisateurs comme vous.

Malgré tout le chemin que nous avons parcouru, je me retrouve encore tous les jours aussi frustré que lorsque le projet WordPress a commencé. Quand nous recevons des prix ou qu'un nouvel excellent livre sort, comme celui que vous tenez entre les mains, cela ne veut pas dire que nous puissions nous reposer sur nos lauriers ; c'est notre défi, notre responsabilité toujours intense, de créer le meilleur logiciel de publication dans le monde, sans comparaison possible. Nous n'y parviendrons pas en faisant de plus grosses dépenses que la concurrence, mais en aimant toujours plus ce que nous faisons. Nous le ferons grâce aux développeurs et aux utilisateurs qui se tiendront par la main. Nous le ferons ensemble.

On dit souvent qu'il faut dix ans pour faire un grand logiciel. WordPress a récemment fêté son cinquième anniversaire – c'est donc que nous sommes à mi-chemin.

Matt Mullenweg
Fondateur du projet WordPress
Août 2008

1 Introduction

Les idées sont faites pour se propager, et aujourd'hui les moyens de propagation n'ont jamais été aussi nombreux, amenant même notre époque à recevoir l'appellation d'ère de la communication. Entre baisse des prix et amélioration des réseaux, Internet autorise aujourd'hui tous ceux qu'une idée anime à la faire entendre plus globalement qu'avec un téléphone, de manière plus abordable qu'avec la télévision, plus rapide qu'avec un livre, plus pérenne qu'avec le bouche-à-oreille… Si vous avez une idée à faire partager, Internet est devenu incontournable.

Au cœur d'Internet se trouve le World Wide Web, fait d'innombrables sites abordant autant de sujets. Et une part toujours plus grande de ces sites web appartient à la catégorie des blogs.

Vous avez ouvert ce livre parce que vous voulez en savoir plus sur la publication en ligne à l'aide de WordPress, CMS spécialisé en la matière.

Petit guide du blogueur débutant

Qu'est-ce qu'un blog ?

Cette introduction est probablement l'occasion idéale pour définir ce qu'est un blog. Sans plonger dans l'explication technique ni la justification "philosophique", on peut simplement dire qu'un blog est une forme particulière de site web, avec des caractéristiques précises, mais qui ne sont pas toujours présentes. La définition la plus large est la suivante : un blog se caractérise par la publication plus ou moins régulière d'articles, généralement classés par ordre chronologique inverse sur la page principale. Les articles sont datés, et chacun dispose d'une URL unique dotée d'une zone de commentaire.

Déjà dans cette définition, certains points sont facultatifs : l'ordre peut ne pas être strictement chronologique, il se peut que les articles n'aient pas d'URL unique, ou l'auteur a pu choisir de refuser les commentaires. Partant de là, de nombreux composants peuvent s'ajouter : présence ou non d'un flux de syndication, utilisation d'un système de gestion de contenus (*Content Management System*, ou CMS), ou écriture à la main du code HTML, etc. Les libertés sont nombreuses, et effectivement il est difficile de donner une définition qui puisse s'appliquer à absolument tous les blogs.

Dans les faits, un blog est davantage défini par son aspect extérieur que par son contenu.

On pourrait ainsi différencier sites web et blogs en précisant simplement que ces derniers sont régulièrement mis à jour. Un site web reste quant à lui statique – même s'il peut inclure une section blog, ce qui est souvent une solution idéale pour faire vivre un site statique…

Il existe toutes sortes de blogs, depuis le blog très personnel axé sur un sujet intime jusqu'au blog d'entreprise servant de média direct entre l'entreprise et ses clients. Certains blogs sont beaucoup lus et même influents auprès des médias traditionnels, tandis que d'autres ne sont lus que par une poignée de personnes proches du blogueur… Politique, promotion produit, couture : tous les sujets sont abordés, et les blogs les plus visités sont souvent ceux qui expriment la passion de leur auteur.

Tenir un blog est aujourd'hui un phénomène reconnu et solidement installé dans les habitudes des internautes. L'influence des blogs est d'ailleurs telle qu'il peut être difficile de différencier blogs et médias traditionnels, tant les journalistes qui tiennent un blog sont nombreux, et tant les caractéristiques du format "blog" sont fréquemment reprises par les sites "classiques". L'expression "avoir un blog" est donc le plus souvent réservée aux personnes qui tiennent un blog personnel, mais l'expression "avoir un site web" est tout aussi valable désormais.

Qu'est-ce que WordPress ?

WordPress est un logiciel de gestion de blog gratuit et open-source – c'est-à-dire que tout le monde peut participer à son évolution. Âgé aujourd'hui de 7 ans, il est en mesure de gérer bien plus qu'un simple blog : il peut être utilisé pour construire un site dynamique complet.

Mené initialement par deux personnes (Matt Mullenweg et Mike Little), le projet WordPress est désormais essentiellement maintenu par les développeurs de la société Automattic, créée par Mullenweg en août 2005, ainsi que par un grand nombre de développeurs indépendants. Sous l'égide d'Automattic, WordPress a évolué pour exister sous plusieurs formes. C'est pourquoi le nom WordPress désigne deux projets distincts, mais aux racines communes :

- **WordPress, le gestionnaire de blog open-source.** Projet le plus connu, WordPress (ou WP) est un script PHP gratuit qui peut gérer un site web complet grâce à ses fonctionnalités avancées de gestion de contenus tels que les pages statiques. Il peut également gérer un réseau de sites (ou "ferme de blogs") très rapidement – bien que cela soit à réserver aux spécialistes, car sa maintenance n'est pas si aisée que pour une installation monosite. Ce projet constitue le sujet principal de ce livre.
- **WordPress.com, le service d'hébergement de blogs.** C'est un service gratuit d'hébergement de blogs, avec options payantes. Géré par Automattic, ce service est basé sur WordPress MU.

Afin de différencier WordPress-le-script de WordPress.com-le-service-d'hébergement, on utilisera régulièrement les appellations "WordPress.org" ou "la version autonome de WordPress". Plus globalement, si le ".com" n'est pas mentionné, on peut sans crainte partir du principe que l'on parle de la version autonome.

À noter qu'une troisième édition de WordPress existait jusque récemment : WordPress MU. Avant sa version 3.0, WordPress ne pouvait gérer qu'un seul site par installation. WPMU était une déclinaison de WP permettant de gérer plusieurs sites. Avec la version 3.0, WPMU a été intégré à l'édition classique de WordPress et n'a donc plus de raison d'être cité. On parle désormais de "WordPress en mode multisite".

Outsider à l'origine, WordPress est aujourd'hui l'un des outils de gestion de blog les plus populaires : les statistiques de 2009 relevaient environ 6 millions de blogs WordPress ; de son côté, le service WordPress.com comptait plus de 4 millions de blogs, plus de 140 000 articles publiés chaque mois en moyenne. Sa popularité dépasse aujourd'hui celle du service gratuit Blogger.com (propriété de Google). En avril 2010, lors d'une conférence autour du CMS Drupal, les résultats d'une récente étude annonçaient que WordPress propulsait 8,5 % de l'ensemble des sites web. En considérant que Google indiquait en 2008 qu'il indexait plus de 100 milliards de sites, cela mettrait le nombre de sites WordPress aux alentours de 8 milliards...

La société Automattic ne fait pas que gérer ces deux incarnations de WordPress : dans son portfolio de projets, on retrouve un outil open-source de gestion de forum (bbPress), un service gratuit de lutte contre le spam (Akismet, inclus par défaut avec WordPress), un service gratuit de gestion des avatars en ligne (Gravatar), un outil open-source de gestion de site communautaire/réseau social (BuddyPress, basé sur WordPress), un service gratuit de prise en charge avancée des commentaires de blog (Intense Debate), un service gratuit de gestion de sondages (PollDaddy), et de nombreux autres projets... Automattic était à l'origine un nom générique pour les projets personnels de Matt Mullenweg, et il est devenu aujourd'hui l'un des acteurs les plus puissants du monde du blog ; il compte plus de 30 employés, et a récemment levé plus de 30 millions de dollars d'investissement afin de consolider ses fondations et de voir à long terme. L'avenir de WordPress semble donc assuré…

Qu'est-ce qu'un bon blog ?

Bien qu'il n'existe pas de règles gravées dans le marbre, la blogosphère est aujourd'hui suffisamment mature et variée pour qu'il soit possible de discerner les points communs des blogs de qualité :

- **Il a une voix personnelle.** Écrire un blog sur rien ne vous amènera pas bien loin, tout comme écrire un blog sur tout ce qui se passe ne vous rendra jamais plus intéressant aux yeux des internautes, qui recherchent avant tout l'originalité. Cette originalité, qui sortira votre blog du lot, ne peut s'établir qu'à partir du moment où vous écrivez sur des sujets qui vous touchent profondément ou vous passionnent réellement. L'idée est de faire "votre trou" dans la blogosphère en écrivant avec passion, passion qui trouvera forcément un écho chez certains blogueurs/internautes, si tant est que vos propos soient sincères.
- **Il encourage la conversation.** Tenir un blog est une chose, c'en est une autre que recevoir des commentaires pertinents, et même voir se former une véritable communauté d'habitués autour du blog. Cette communauté ne se formera pas du jour au lendemain : il faut non seulement interpeller ses lecteurs (sans pour autant finir chaque article par "qu'en pensez-vous ?"), mais également s'inscrire dans la communauté plus grande de la blogosphère, à savoir laisser des commentaires appropriés sur les autres blogs (ce qui par ailleurs vous ramènera des visiteurs), mais encore répondre aux articles des autres blogs par vos propres articles bien construits. Bien sûr, cela demande du travail et n'est

nullement nécessaire pour être heureux avec son blog : certains blogs n'autorisent pas les commentaires et n'en sont pas moins appréciés.

- **Il sait rester humble.** Votre avis n'est pas toujours plus important ou plus judicieux que celui des autres, et si jouer la provocation peut vous rapporter un lectorat, il sera bien moins fidèle que celui d'un blog sobre et affable. Nombreux sont ceux qui se lancent dans la blogosphère avec pour objectif de s'y faire un nom, voire de devenir un blogueur "influent" (ou un "influenceur", un "buzzeur", *ad nauseam*, etc.). L'idée du blog est d'établir une conversation, pas de convaincre de putatifs lecteurs du bien-fondé de vos arguments.
- **Il respecte ses lecteurs.** Cela suppose de ne pas modifier les articles déjà publiés afin d'en arranger le propos sans l'indiquer clairement, de ne pas effacer les articles que l'on regrette (sauf injonction judiciaire), et de toujours publier les commentaires des lecteurs (sauf cas de force majeure). Avoir une étiquette de publication est essentiel pour établir une relation de confiance avec ses lecteurs.
- **Il respecte le contenu d'autrui.** Un blog doit faire preuve d'une certaine originalité pour sortir du lot. Certains blogs se sont fait une spécialité de reprendre mot pour mot les contenus d'autres sites (images, vidéos, parfois même textes) afin de surfer sur l'éphémère vague de "buzz" que ce contenu peut générer, et profiter des possibles requêtes Google sur le sujet – et, donc, espérer des clics sur leurs bannières publicitaires. Si ces blogs sont certes populaires (dans le sens le moins noble du terme), cela ne présage pas de leur qualité : ils n'apportent rien à la conversation, n'étant qu'un relais vide de sens. Comme pour tout média, le droit de citation existe sur les blogs, et l'exigence de qualité ne doit amener la (courte) citation que pour mieux la discuter, non pour avoir sur son blog les mêmes mots-clefs que sur tant d'autres blogs ciblés "buzz". Plus prosaïquement, le contenu d'un site étant une œuvre de l'esprit, il entre dans le cadre de la propriété intellectuelle. De fait, reprendre un contenu sans autorisation explicite de l'auteur peut s'apparenter à du piratage…
- **Il est libre de toute contrainte.** Terminons cette liste en tempérant les points précédents : un bon blog prend ses lecteurs au dépourvu, il sort de sa propre "ligne éditoriale" (si tant est qu'il en ait une), il se met ses lecteurs à dos et les fait réagir – idéalement, en les poussant à bloguer de leur côté plutôt qu'écrire un commentaire. Un bon blog doit donner envie d'ouvrir un blog.

En définitive, la caractéristique première d'un bon blog reste sans doute sa liberté de ton, aussi les points que nous venons d'évoquer ne sauraient être des vérités scientifiques, mais sont plutôt des "bonnes pratiques" ; ne pas les suivre à la lettre ne fera probablement pas de vous un mauvais blogueur. Soyez libre !

Faire ses premiers pas dans la blogosphère

La blogosphère est une communauté très ouverte : il est extrêmement facile d'y entrer, il suffit de créer un blog. En revanche, il est plus difficile d'y rester, de s'y faire remarquer ou même de s'en faire oublier, de comprendre ses dynamiques et ses politiques internes…

Devenir membre de la blogosphère, pour celui qui souhaite s'y mettre sérieusement, requiert plus que quelques pensées matinales et des photos de chat en contre-jour.

Si vous vous apprêtez à ouvrir votre premier blog, gardez les points suivants en tête, ils vous éviteront un certain nombre de faux pas !

- **Tout le monde vous lit.** Votre blog sera accessible à tous, notamment aux moteurs de recherche. Évitez donc d'écrire des textes que vous ne souhaiteriez pas voir ressurgir dans quelques années. En règle générale, si vous écrivez sous votre nom, il peut être bon de bloguer comme si vous vous adressiez à vos parents, votre grand-mère, votre patron, vos anciens collègues – qui tous, potentiellement, peuvent vous lire. Et, si vous écrivez sous un pseudonyme, ne croyez pas être anonyme pour autant…
- **L'anonymat n'existe pas vraiment sur Internet** et n'offre donc qu'une protection toute relative. Plus d'un s'est brûlé les ailes en se croyant en sécurité. À tout prendre, il est même préférable de bloguer sous votre véritable nom, afin de contrôler vous-même votre image sur Internet, plutôt que risquer de voir une recherche sur votre nom renvoyer vers un site que vous ne contrôlez pas. Le blog y perdra sans doute en liberté de ton, mais votre réputation n'en sortira certainement que grandie.
- **Gardez vos secrets, et ceux des autres.** Un blog est un "journal intime pas intime" en ceci que vous pouvez y raconter la vie de votre entourage et que son contenu devient aussitôt public et accessible à tous. Ne parlez donc que de ce qui vous implique vous, et non une autre personne. En général, évitez de citer les noms complets – surtout s'ils n'ont pas déjà un site web officiel, car votre article pourrait alors être le premier résultat d'une recherche sur le nom !
- **Les moteurs de recherche archivent tout, pour toujours.** Les requêtes des moteurs de recherche se font non pas directement sur Internet mais sur une copie de votre site qui est régulièrement faite et stockée sur les serveurs de Google, Yahoo!, Archive.org et tant d'autres. Si vous effacez un article ou fermez le blog, il existera toujours sur les serveurs de ces moteurs, et il ne sera pas évident de l'en supprimer. Soyez donc bien conscient de l'incontrôlable pérennité de vos écrits sur Internet et des implications possibles pour vous ou votre entourage dans un an, cinq ans, dix ans…
- **Respectez les codes de conduite établis.** La blogosphère est très permissive, mais les règles de base de l'échange sur Internet – et dans la "vraie vie" – s'y appliquent toujours. Comme dans toute culture, il existe des codes sociaux, et ce n'est pas parce qu'on est en ligne qu'il faut traiter les gens différemment. La Netiquette, imaginée bien avant la création du Web, est une charte de règles de conduite et de politesse qui tente de décrire le comportement à adopter pour que les relations sur le Net restent courtoises. Sans que la Netiquette soit absolue ou définitive, il faut garder en mémoire que sur Internet la personne en face reste un être humain et que les querelles de clocher n'apportent rien à l'idéal humaniste que représente ce réseau mondial.

Ce ne sont là que quelques règles de base, qui traduisent la bienséance du réseau. En définitive, vous restez seul maître de vos écrits : il n'y a pas et il n'y aura jamais de "déontologie universelle du blogueur".

Le blogueur est donc seul maître à bord, mais également seul responsable devant autrui et devant la loi – ce qui forme un autre thème important, auquel nous consacrons la section suivante.

Responsabilité du blogueur

Si Internet est un vaste lieu de liberté, les internautes n'en sont pas moins soumis aux lois terrestres, qui s'appliquent d'autant plus aux blogueurs, étant donné que leurs sites sont régulièrement mis à jour, et donc très favorisés par les moteurs de recherche. Depuis le début, la justice surveille cet espace, et des blogs se sont déjà fait condamner à cause d'un article qu'ils publiaient, ou même d'un commentaire qu'ils affichaient. La prudence reste donc de mise.


Attention

Les auteurs n'étant pas juristes, nous avons tiré une grande partie du contenu de cette section d'un article publié par l'avocat-blogueur maître Eolas, article que nous vous encourageons à lire : **http://maitre-eolas.fr/2008/03/24/905-blogueurs-et-responsabilite-reloaded**.
Par ailleurs, cette section ne fait référence qu'aux lois applicables en France. Nous demandons aux lecteurs francophones d'autres pays (Suisse, Belgique, Québec, etc.) de bien vouloir nous excuser de ne pas avoir pu prendre en compte leurs lois locales, et les enjoignons à se renseigner.
Enfin, notez que cette section se base en majeure partie sur la législation de 2008, et ne prend pas forcément en compte les dernières évolutions de la loi en la matière.

Légalement, c'est l'éditeur du site qui est responsable de son contenu, et non l'hébergeur du site, comme d'anciennes affaires judiciaires peuvent le faire croire. Cette responsabilité a été établie par la loi pour la confiance dans l'économie numérique (LCEN), et la jurisprudence n'a de cesse d'aller dans ce sens.

Un blog étant un site web à part entière, le blogueur est censé respecter certaines obligations :

- **Déclarer son identité à son hébergeur.** Un hébergement sous un faux nom, même gratuit, est un délit.
- **Faire figurer sur le site le nom du responsable**, ou, en cas de site non professionnel et anonyme, la mention de l'hébergeur qui a les coordonnées du responsable. Ce texte se trouve souvent placé dans la rubrique Mentions légales.
- **Publier gratuitement un droit de réponse** de toute personne nommée ou désignée dans un article ou un commentaire, sous trois jours à compter de la réception.

Il faut bien comprendre qu'un blog entre dans le cadre du droit pénal de la presse et de l'édition, à l'instar de tout écrit mis à la disposition du public, et celui du droit à l'image et de l'intimité de la vie privée. La liberté d'expression perd donc ses droits devant un texte diffamatoire, faisant l'apologie des crimes contre l'humanité, incitant à la haine raciale, contenant de la pornographie infantile, poussant à commettre des crimes ou délits.

Si la plupart de ces actes sont heureusement rares, il est cependant facile d'écrire un texte injurieux ("expression outrageante ne contenant l'imputation d'aucun fait") ou diffamatoire ("allégation ou imputation d'un fait qui porte atteinte à l'honneur ou à la considération de la personne ou du corps auquel le fait est imputé") sans forcément se rendre compte des

implications – surtout si le propos diffamatoire provient d'un commentaire, dont le blogueur est légalement responsable. Les blogueurs peuvent le plus souvent se reposer sur la prescription, qui s'applique trois mois après la publication de l'article ou du commentaire en cause – mais qui pourrait passer à un an si une récente proposition de loi venait à passer.

Le respect de la vie privée fait également partie intégrante de la responsabilité du blogueur : l'article 9 du Code civil français est sans ambiguïté, "chacun a droit au respect de sa vie privée", et cela s'applique aussi bien aux voisins ou amis qu'aux personnes publiques. Une star de cinéma peut donc légalement se plaindre si vous publiez des photos de sa vie privée…

La loi peut même s'appliquer à un blogueur n'ayant pas commis de délit particulier : des sanctions professionnelles ont déjà été prises à l'égard d'employés ou de fonctionnaires parce qu'ils écrivaient pour leur blog pendant leurs heures de travail, qu'ils divulguaient des informations confidentielles, ou simplement qu'ils parlaient de leurs professions alors qu'ils devaient s'en abstenir – et c'est ici qu'il faut bien comprendre que l'anonymat n'existe pas sur Internet.

Les blogueurs américains ont même une expression pour cela, *to get dooced*, tirée du pseudonyme de Heather Armstrong, qui en 2002, un an après avoir ouvert son blog à l'adresse **http://www.dooce.com**, a été renvoyée de sa société pour avoir parlé de ses collègues de manière satirique.

Plus proche de nous, la blogueuse anonyme derrière le site **http://www.petiteanglaise.com/** a été renvoyée pour "faute grave" en 2006, après avoir blogué sur l'ambiance à son travail – l'employeur n'était pourtant jamais identifiable dans les articles, qui ne nuisaient donc pas à l'entreprise. Par ailleurs, le blogueur Garfieldd a été révoqué de l'Éducation nationale en 2006 pour avoir tenu un blog sur sa vie de proviseur et d'homosexuel – sans jamais mélanger les deux ni faire de rapprochement qui puisse tomber sous le coup de la loi pénale.

Certes, ces deux affaires françaises se sont plutôt bien terminées : Petite Anglaise a ensuite gagné son procès devant les Prud'hommes et a écrit des livres ; Garfieldd de son côté a vu sa révocation ramenée à une exclusion pendant un an et une mutation ; mais chaque affaire est unique…

Ajoutons enfin un rapide laïus à propos des commentaires. En effet, deux possibilités se présentent :

- **Le commentaire n'est pas en ligne tant que vous ne l'avez pas validé.** Vous donnez donc un accord explicite à ce commentaire, et votre responsabilité est totalement engagée. Valider les commentaires manuellement revient à en devenir l'éditeur, voire l'auteur.
- **Le commentaire est automatiquement validé.** La publication se fait donc sans votre accord, et votre responsabilité est moindre – mais pas nulle pour autant. Vous êtes considéré comme un hébergeur.

Dans tous les cas, vous devez faire attention autant au contenu de vos articles qu'à celui de vos commentaires...

Ce n’est là qu’une approche très succincte de ce thème, que l’on pourrait résumer ainsi : prenez garde à ce que vous publiez sur votre blog. Ce que vous y écrivez pourrait revenir vous hanter dans plusieurs années, soyez donc certain d’assumer totalement, et sur le long terme, vos propos.

Découvrir la culture blog

Les origines des blogs

Le monde du blog tel que nous le connaissons aujourd’hui est le résultat d’années d’évolutions, de découvertes et de formalisation. Il est toujours intéressant de se plonger dans ce passé pour comprendre ses conséquences sur la blogosphère actuelle.

Pour commencer, il faut bien savoir que dès sa création le Web a été pensé comme un média où les internautes pouvaient aussi bien lire qu’écrire – le premier navigateur était d’ailleurs une combinaison de lecteur et d’éditeur de sites web. L’idée première de Tim Berners-Lee, inventeur du World Wide Web en 1990, était que quiconque circulant sur le Web disposerait d’un espace personnel sur lequel il pourrait écrire ce qu’il veut, simplement – d’où l’éditeur intégré au navigateur et accessible par un simple clic.

Les évolutions logicielles ont séparé navigateurs et éditeurs de pages web (exception faite d’Amaya), mais l’avènement des blogs et des wikis a permis à la vision initiale de Tim Berners-Lee de se réaliser pleinement : il est aujourd’hui possible, d’un simple clic, de publier ses idées en ligne.

Le mouvement de la communauté "blog" n’a pas eu une naissance spontanée ni une popularité immédiate, mais trouve probablement ses origines dans les BBS, ces machines hébergées chez des particuliers, administrées par des "sysop" (system operators), reliées au réseau téléphonique commuté, et auxquelles les utilisateurs pouvaient se connecter par le biais d’un appel téléphonique au travers d’un modem. Sur ces BBS se sont créées les premières communautés numériques, forcément disparates mais permettant de partager des fichiers autrement plus facilement qu’en envoyant des disquettes par la poste, comme c’était l’usage au sein des communautés de pirates informatiques et de créateurs de démos.

De simples dépôts de fichiers, avec de strictes règles de quotas d’envoi et réception de fichiers, les BBS ont rapidement évolué pour inclure des systèmes de messagerie interne et de forums dédiés aux divers sujets sur lesquels le BBS était spécialisé, jusqu’à devenir de simples endroits où les membres pouvaient discuter de tout et n’importe quoi…

En France, les BBS peuvent être comparés aux serveurs RTC, qui exploitaient les possibilités graphiques et réseau du Minitel. Partage de fichiers, forums de discussion, graphisme, gestion par des particuliers passionnés : BBS et serveurs RTC ont beaucoup en commun.

Les BBS sont apparus dans les années 1980, et leur popularité n’a cessé d’augmenter avec l’amélioration des modems, culminant vers la fin des années 1990, au moment où le Web est devenu accessible à tous, notamment avec des offres avantageuses comme celles d’AOL.

L’ouverture au grand public des outils d’Internet, notamment Usenet (les *newsgroups*, ou groupes de discussion), du courrier électronique et l’apparition des listes de diffusion

(mailing-list) et enfin du Web avec la naissance des forums de discussion, tout cela a participé à montrer qu'il était possible de publier ses idées directement, depuis un ordinateur.

Partant de là, les premiers sites web personnels ont mené à la publication en ligne de journaux personnels, et donc au passage au numérique du diarisme – d'où une définition du terme blog le décrivant comme un "journal intime pas intime", voire un "journal extime"…

Apparition et popularité des blogs

Le prix du premier blog revient sans doute à l'inventeur du Web, Tim Berners-Lee, qui en 1992 avait mis en place une page "What's New" permettant de suivre les évolutions de son projet World Wide Web : nouvelles versions de navigateurs, nouveaux serveurs, nouveaux membres de l'équipe.

Les premiers diaristes du Web, ces particuliers qui se sont lancés dans la description régulière et sur le Web de leur quotidien et/ou de leurs pensées, sont apparus en 1994. Le plus célèbre d'entre eux est Justin Hall, un étudiant américain qui en janvier 1994 s'est mis à lister et à commenter ses bons liens du moment au fur et à mesure qu'il les découvrait, puis, en janvier 1996, s'est mis à documenter sa vie intime sur son site links.net (voir Figure 1.01). Il a tenu son blog jusqu'en 2005.



Figure 1.01
Links.net en décembre 1996.

Dans les pays francophones, c'est le Québec qui tire le premier avec Montréal, soleil et pluie de Brigitte Gemme, de juin 1995 à juin 1998, après quoi il faudra attendre août 1999 pour voir l'arrivée du blog collaboratif anonyme Pssst.ca.

On l'aura compris, l'essor des blogs n'a pas été immédiat : tout comme une simple page mise à jour à la main suffisait à lister les nouveaux sites web en 1993, il existait une page recensant les nouveaux blogs, tenue à la main par le blogueur Jesse James Garrett. En janvier 1999, celle-ci en recensait 23. Mais déjà la communauté s'était formalisée, d'une certaine manière, en adoptant le terme "weblog", créé par John Barger en décembre 1997 sur son blog robotwisdom.com – contraction de *web log*, littéralement "journal en ligne" –, qui décrivait le fait de "tenir le journal de son activité sur le Web" (*logging the Web*). Petit à petit, les sites personnels régulièrement mis à jour par leurs auteurs ont été décrits comme des blogs…

Weblog est devenu "blog" suite à une plaisanterie du développeur et blogueur Peter Merholz, qui écrivait en mai 1999 : "Pour ce que ça vaut, j'ai décidé de prononcer le mot weblog comme [wee'-blog]. Ou *blog* pour faire court."

L'évolution de weblog en blog aurait pu rester un simple clin d'œil entre connaisseurs, si quelques mois plus tard, en août 1999, la société Pyra Labs n'avait sorti l'outil en ligne Blogger.com, une interface complète et gratuite de mise à jour de site (voir Figure 1.02). Ce n'était certes pas le premier outil de gestion de blog (Open Diary avait été lancé en octobre 1998, LiveJournal, en mars 1999), mais la simplicité d'utilisation de Blogger fera que ce dernier aura un impact important sur le monde de la publication personnelle en ligne ; impact qui, accessoirement, aidera le mot "blog" à entrer dans le langage courant des internautes.



Figure 1.02
Blogger.com en janvier 2000.

Les quelques mois qui ont précédé le lancement de Blogger.com ont connu une explosion : d'abord amorcée par une poignée d'adopteurs précoces (*early adopters*) qui souvent se connaissaient, puis les suiveurs ont commencé à lancer leurs propres blogs, enfin l'arrivée de services simples et gratuits comme Blogger.com a véritablement favorisé un déluge de blogs sur Internet. En simplifiant la publication en ligne au point de ne plus devoir écrire

de code HTML ni utiliser de client FTP, les services en ligne comme Blogger.com ont véritablement popularisé ce type de site.

À partir de cette ouverture au plus grand nombre, la blogosphère n'a fait que grandir en volume et en diversité, les outils se sont multipliés, la couverture médiatique s'est affolée…

Au niveau politique, l'influence des blogs a commencé véritablement à se faire sentir après les attentats du 11 septembre 2001 et dans le climat politique tendu aux États-Unis après cette date : de nombreux blogueurs se sont lancés, et les médias se sont enfin penchés sur le mouvement. Entre préparatifs de guerre, limitation des droits individuels et unilatéralité des médias, les blogs ont pu donner une opinion alternative de plus en plus forte et omniprésente, allant jusqu'à dénoncer les manquements des politiciens comme des journalistes, avec des conséquences très réelles – démission du sénateur Trent Lott, du journaliste Dan Rather… Dès les élections présidentielles américaines de 2004, les blogueurs étaient traités (bien que de manière expérimentale) comme des journalistes aux conférences.

Désormais, toute personne souhaitant toucher le grand public (du moins ceux ayant accès à Internet) se doit de tenir un blog. Politiciens, chefs d'entreprise ou simple quidam : le blog autorise une immédiateté et une proximité jamais atteintes auparavant, mettant toutes les couches sociales sur un pied d'égalité – celui de la libre publication sur Internet. Nombreux sont ceux qui voient en Internet le plus grand progrès pour l'humanité depuis l'invention de l'imprimerie, et les blogs ne font que simplifier l'exploitation de ce progrès…

Et dans les pays francophones…

Comme nous l'avons signalé auparavant, le premier blog francophone date de 1995, mais il faudra attendre 1999-2000 pour voir les francophones se lancer en masse, avec des blogs qui sont encore ouverts et actifs aujourd'hui.

La blogosphère francophone a suivi le même chemin que la blogosphère américaine, avec quelques années d'écart : elle dispose de "stars", de communautés, de politiciens et de journalistes, de son propre classement de popularité, et organise même des réunions mensuelles (ParisCarnet à Paris, YULblog à Montréal, Yulbuz à Québec, Strasbourg et Bruxelles, Bloggy Friday à Lausanne, etc.) où les blogueurs se rencontrent, le plus souvent dans un bar.

Les politiques français ont assez vite compris l'intérêt d'avoir une tribune libre sans devoir passer par les médias traditionnels. Alain Rousset, candidat aux élections régionales en Aquitaine, ouvre le bal en décembre 2003, rapidement suivi par Dominique Strauss-Khan et Jean-François Copé en février 2004.

Parmi les personnages en vue de la blogosphère francophone, on peut sans doute nommer Loïc Le Meur (**http://www.loiclemeur.com**). Entrepreneur infatigable, il s'intéresse aux blogs au moment où les médias commencent à se pencher sur le sujet, en septembre 2003, et très rapidement rachète la plate-forme U-Blog, créée par Stéphane Le Solliec, afin de monter la société du même nom. Dès le mois de mars 2004, U-Blog devient distributeur exclusif des produits Six Apart (Movable Type, TypePad), et Loïc Le Meur devient le premier promoteur des blogs en France. Par son action, puis par un effet d'imitation, nombreux sont les

chefs d'entreprise, grands médias et politiciens qui se lanceront dans la blogosphère. En ce sens, Loïc Le Meur a eu une grande influence dans la formation de la blogosphère française actuelle, recevant même le titre de "pape des blogs". Depuis, il est plus discret, mais ses quatre années de promotion des blogs dans les médias ont laissé une certaine trace.

Longtemps avant l'arrivée inéluctable de "VRP" comme Loïc Le Meur, la promotion des blogs revenait aux *early adopters*, le plus souvent des *geeks* (ou affiliés) qui n'avaient pas peur des nouvelles idées. Par ironie, ceux qui avaient ouvert un blog avant Loïc Le Meur (donc, avant le 29 septembre 2003) sont parfois surnommés "dino-blogueurs" – à différencier des "néo-blogueurs", apparus après Loïc Le Meur et ne rechignant pas devant les articles sponsorisés (équivalent du publireportage sur Internet), les encarts publicitaires et le marketing personnel.

Parmi ces pionniers, il faut noter le travail de la Suissesse Stephanie Booth (**http://climb-tothestarscom**), qui sans être totalement geek (elle a fait des études de lettres) ni totalement francophone (elle est d'origine anglaise) a été la fondatrice ou partie prenante de nombreux projets fondamentaux pour les blogs en particulier et les standards du Web en général. Ainsi, elle a lancé dès 2001 le wiki Spirolattic pour la promotion en français de ses sujets de prédilection, puis le site Pompage.net pour traduire les articles anglais sur la conception web. Étant bilingue, elle a très tôt lancé des débats sur la terminologie des blogs, et a participé à la promotion des blogs au sein de la communauté des développeurs.

Ils sont nombreux à avoir très tôt défriché le terrain des blogs en francophonie, au profit de tous. Sans être exhaustifs, nous pouvons citer ceux qui sont encore actifs aujourd'hui, comme Laurent Gloaguen (**http://embruns.net/**), Karl Dubost (**http://www.la-grange.net/**), ou encore Delphine Dispa (**http://www.oeildemouche.net/**).

La blogosphère d'aujourd'hui

La blogosphère fonctionne toujours sur les bases établies en 1999 par la montée en popularité des blogs. Selon l'outil de mesure de blogs Technorati, le nombre de blogs suit une courbe quasi exponentielle, depuis les premières mesures réalisées en mars 2003. En avril 2007, leur nombre était estimé à 70 millions, avec environ 120 000 blogs créés chaque jour, soit 1,4 chaque seconde. D'après les statistiques de l'outil, le nombre de blogs double tous les cinq à sept mois.

Les communautés francophones sont fortes et ne se mélangent pas toujours. La plus importante est bien entendu celle des blogs Skyrock, qui regroupe plus de 19 millions de blogs, ce qui en fait le plus gros réseau social de France, devant Facebook. La communauté des blogueurs adolescents forme donc une part importante de la blogosphère, mais la communauté des geeks est également très forte, tout comme celles, inattendues, des fans de tricot ou de scrapbooking, très soudées et dont le nombre est insoupçonnable.

Les blogs font désormais partie du paysage familier d'Internet – ils sont même trop présents pour certains, qui préféreraient que des blogs personnels ne surgissent pas en trop haute position pour certains résultats clés des moteurs de recherche. En effet, ces derniers ont tendance à donner plus de poids aux sites régulièrement mis à jour ; donc, pour un même sujet, les blogs sont souvent gagnants face à un site statique – autre avantage d'utiliser un blog.

Cette forme de publication s'est popularisée très rapidement, et il est impossible de limiter les blogs à une catégorie de gens tant ils sont simples d'utilisation : qu'on soit geek, politicien, artiste (dessinateur, musicien, etc.), ou tout simplement désireux de partager, toutes les passions se retrouvent sur Internet.

Les entreprises elles-mêmes ont bien compris l'intérêt de ce nouveau média, et lancent toutes sortes de blogs : communication interne ou avec la clientèle, promotion d'un produit… Mais il faut accepter de voir ses erreurs pointées du doigt par les lecteurs, ce qui n'est pas forcément donné aux sociétés qui découvrent ce nouveau moyen de diffusion.

Le média blog s'est diversifié pour accueillir plus que du texte et des liens : les photographes présentent leurs travaux sur un photoblog, les vidéastes ont leur vidéoblog, et il est possible de gérer à moindres frais son propre média audio ou vidéo en produisant un podcast. Cette dernière appellation décrit un blog (professionnel ou personnel) proposant régulièrement en téléchargement des émissions audio ou vidéo – chose qui était inimaginable il y a quelques années encore…

Les blogs sont par ailleurs devenus sources de profits pour certains. Beaucoup de blogueurs ont mis en place des publicités (le plus souvent textuelles) sur leurs pages, en laissant le soin à de jeunes régies de gérer les publicités. Google profite ainsi de cette opportunité avec sa régie AdSense, Yahoo! avec son Publisher Network, Microsoft avec adCenter, et un grand nombre de petits acteurs, comme Text-Link-Ads, qui cherchent à exploiter cette nouvelle manne. Certains blogueurs y trouvent un apport léger, mais une poignée peut se permettre de vivre uniquement de leur blog – parfois même en se faisant payer pour publier des articles. Ce dernier point provoque les polémiques habituelles d'un mouvement alternatif dont la notoriété explose : opposition entre pionniers et marchands, entre défricheurs et influenceurs, entre humour de connivence et déballage exclusif du dernier téléphone à la mode…

Les blogs font aujourd'hui partie intégrante du paysage des internautes et deviennent donc le reflet de notre société, où l'excellence croise l'imperfection – et parfois partagent le même hébergeur…

Les différents systèmes de blog

WordPress est loin d'être la seule solution lorsqu'on souhaite se lancer dans la publication en ligne. Depuis l'avènement de cette forme de site, un trop grand nombre de logiciels et outils ont été créés pour qu'on puisse imaginer en dresser une liste exhaustive.

Bien sûr, nous pensons que WordPress est en mesure de répondre à toutes les attentes d'un blogueur, mais il est toujours bon de savoir dans quoi on s'engage et ce qui existe ailleurs. Aussi, nous allons prendre le temps de parler des autres systèmes de blogs. Ceux-ci se regroupent en deux principales catégories : les services d'hébergement de blogs, et les logiciels à héberger soi-même.

La sélection qui suit n'aborde que les systèmes les plus réputés. Tous ne sont pas gratuits ni open-source, mais chacun dispose d'une large base d'utilisateurs et d'un développement soutenu, ce qui leur assure une certaine pérennité.

Services d'hébergement de blogs

Nous allons pour cette section faire une séparation : les services créés par des équipes francophones d'un côté, et ceux d'origine anglophone de l'autre. Dans les faits, la plupart des services anglophones disposent d'une interface en français mais, la traduction étant parfois aléatoire, il peut être préférable de se reposer sur un service francophone dès le départ.

Les francophones

- **Blogspirit.** Service payant lancé en 2004. Il dispose de deux offres payantes, selon les envies du blogueur.
- **Canalblog.** Service gratuit lancé en 2003.
- **Hautetfort.** Service gratuit, lancé en 2003 par Benoît Desavoye et racheté par Blogspirit l'année suivante. Hautetfort reste l'un des services francophones les plus en vue. Dispose d'offres payantes.
- **Joueb.** Site communautaire gratuit et centré autour des blogs de ses membres, lancé en 2001 par Stéphane "Biz" Gigandet, l'un des premiers blogueurs francophones. Joueb est la contraction de "journal web" et est l'une des traductions possibles de "weblog". Dispose de deux offres payantes.
- **Over-blog.** Service gratuit lancé en 2004, et aujourd'hui l'une des principales plates-formes de blog françaises. Il a la particularité de proposer une rémunération en droits d'auteur pour les blogueurs publiant du contenu original et de qualité.
- **Skyrock blog.** Plate-forme gratuite lancée par la radio Skyrock, et connaissant un énorme succès auprès des adolescents. C'est statistiquement le plus gros réseau social francophone, et le plus gros site français respectueux des standards du Web.
- **ViaBloga.** Service payant lancé en 2004 par les créateurs de Joueb.com et disposant de trois offres graduelles. ViaBloga est géré par l'association à but non lucratif du même nom.

Les anglophones

- **Blogger.** L'un des plus anciens services gratuits toujours en ligne, Blogger a été lancé en 1999 par Pyra Labs et ses fondateurs, Evan Williams et Meg Hourihan. Blogger était un projet annexe mené par Pyra pendant le développement de leur projet principal, un outil de gestion de projet, de contacts et de tâches. Le projet principal a vite été abandonné devant le succès de Blogger. Conçu à l'origine pour mettre à jour un fichier sur un site distant, Blogger est devenu hébergeur à part entière avec le lancement de Blogspot.com ; il est toujours possible de choisir entre les deux modes. Après de nombreuses difficultés financières, le service a été racheté par Google en 2003 et continue d'être développé aujourd'hui. Son cofondateur, Evan Williams, a depuis créé le service de microblogging Twitter.
- **Facebook.** Réseau social lancé en 2004 par Mark Zuckerberg, alors qu'il était étudiant, et aujourd'hui l'un des sites les plus connus sur le Web, avec plus de 100 millions d'utilisateurs actifs. Les membres peuvent écrire des articles et les rendre disponibles *via* un flux RSS.

- **LiveJournal.** Communauté virtuelle gratuite centrée autour du blog, lancée en 1999 par Brad Fitzpatrick, et l'une des plus grandes au monde. Rachetée par la société Six Apart en 2005, LiveJournal a été revendue dès 2007 à SUP, une société russe, la marque LiveJournal étant très populaire en Russie. Elle est gérée par la société LiveJournal, Inc., propriété de SUP, dont Brad Fitzpatrick a rejoint le conseil consultatif. Chaque membre a la possibilité de créer un blog accessible depuis sa page.
- **MySpace.** Réseau social lancé en 2003 par la société eUniverse, spécialisée dans le marketing sur Internet. Racheté en 2005 par le magnat des médias Rupert Murdoch *via* son groupe de médias News Corporation, déjà propriétaire de nombreux journaux et magazines, des studios 20th Century Fox, de la chaîne Fox News, etc. Les utilisateurs peuvent tenir un blog et le diffuser *via* un flux RSS.
- **TypePad.** Service payant lancé par Six Apart en octobre 2003 et aujourd'hui l'un des plus importants hébergeurs payants de blogs. TypePad est basé sur le code de Movable Type, autre produit de Six Apart, et dispose de nombreuses offres payantes.
- **Vox.** Communauté virtuelle gratuite centrée autour des blogs, lancée en 2006 par Six Apart. Vox combine le meilleur de LiveJournal avec tout un espace communautaire très fort.
- **Windows Live Spaces.** Communauté virtuelle gratuite centrée autour des blogs, lancée en 2004 par Microsoft, initialement sous le nom de MSN Spaces.
- **WordPress.com.** Service gratuit lancé par Automattic en 2005, initialement sur la base de WordPress MU, la version multiblog de WordPress. Dispose d'une offre payante pour les blogueurs demandeurs.

Logiciels à héberger soi-même

Les services d'hébergement facilitent grandement la maintenance d'un blog, mais le blogueur qui veut voir par lui-même s'y trouve rapidement limité. Les logiciels à héberger soi-même permettent de maîtriser totalement l'outil : nom de domaine, apparence, extension, taille et nombre des fichiers joints, etc. En contrepartie, le blogueur devra gérer tout seul la maintenance technique du blog…

Notez que, sans prétendre à l'exhaustivité, nous nous efforçons de mentionner tous les logiciels les plus marquants.

- **b2evolution.** Logiciel conçu à partir du code de b2, tout comme WordPress. Créé en 2003 par François Planque, il a dès le départ choisi le support de multiples blogs à partir d'une même installation, ce que WordPress a mis deux ans à mettre en place avec sa version WordPress MU. b2evolution est aujourd'hui toujours en développement, et est utilisé par la plate-forme NRJBlog. Licence open-source.
- **Dotclear.** Logiciel équivalent en de nombreux points à WordPress, conçu par Olivier Meunier en 2003 et très populaire en France. Après une période d'inactivité qui a fait craindre la fin du projet, celui-ci a été remis sur les rails notamment grâce au registrar

Gandi, qui a financé le développement de Dotclear 2.0, une version multiblog sur laquelle est conçue la plate-forme gratuite GandiBlogs. Les développeurs de Dotclear se sont depuis affranchis de Gandi, et le développement a gagné en indépendance grâce à la forte communauté de ce logiciel. Licence open-source.

- **Drupal.** Logiciel de gestion de communauté à la base, Drupal permet de créer toutes sortes de sites, donc des blogs. Conçu en 2001 par le Belge Dries Buytaert, Drupal était à l'origine un système de gestion de forum, mais sa grande extensibilité lui a permis d'évoluer en une véritable plate-forme de publication très complète, et gratuite. Licence open-source.
- **ExpressionEngine.** Logiciel de publication très complet, lancé par Rick Ellis en 2004, et successeur du blogiciel pMachine du même Rick Ellis. C'est un système robuste, professionnel, et doté de nombreuses fonctionnalités. Il dispose d'une offre gratuite, ainsi que de son propre service payant d'hébergement. Licence propriétaire.
- **Habari.** Lancé en janvier 2007 par des designers et des développeurs mécontents de l'évolution de WordPress, Habari se base sur PHP 5 et MySQL 5 plutôt que rester compatible avec PHP 4 et MySQL 4, comme WordPress – ce qui lui permet d'être à la pointe de la technologie. Actuellement en version 0.5.1, il est cependant déjà stable et mature. Licence open-source.
- **Movable Type.** Logiciel de publication lancé en 2001 par le développeur Ben Trott, initialement pour gérer le blog de sa compagne Mena Trott et s'occuper pendant qu'ils cherchaient un emploi tous les deux. Le succès fut tel qu'ils créèrent la société Six Apart dès l'année suivante (le nom désignant le nombre de jours séparant la naissance des deux fondateurs). Six Apart est aujourd'hui l'un des principaux acteurs de la blogosphère, et Movable Type compte parmi les grands concurrents de WordPress. Licence propriétaire, dispose depuis peu d'une version limitée open-source (la question de la licence de Movable Type a longtemps été épineuse, ce qui a bien profité à WordPress).
- **Serendipity.** Surnommé "s9y" et programmé en utilisant PHP et MySQL. Lancé en octobre 2003, ce logiciel qui promettait beaucoup – ses développeurs étant des membres éminents de la communauté – a cependant rapidement été dépassé en popularité par WordPress. s9y a introduit le concept d'extension avec site web centralisé, repris par WordPress. Licence open-source.
- **TextPattern.** Logiciel de publication lancé en 2001 par le développeur Dean Allen. Ayant la particularité de gérer au mieux les CSS et la sémantique de son code généré (notamment par le biais de la syntaxe Textile) en plus d'être très intuitif, TextPattern est considéré comme un logiciel de grande qualité, mais surpassé par les autres offres. Il reste l'une des inspirations initiales de WordPress, au même titre que le défunt logiciel GreyMatter. Licence open-source.
- **Typo.** Logiciel lancé en 2005 par Tobias Lütke, aujourd'hui maintenu par le Français Frédéric de Villamil. À ne pas confondre avec le CMS TYPO3. Licence open-source.
- **WordPress.** Logiciel lancé en 2003 par Matt Mullenweg et Mike Little. Voir la section ci-après pour plus de détails. Licence open-source.

Présentation de WordPress

Prérequis techniques

WordPress est un logiciel écrit en PHP et stockant ses données dans une base MySQL. Les seules conditions qu'un espace d'hébergement doit remplir pour faire fonctionner normalement WordPress sont donc les suivantes :

- PHP en version 4.3 (ou une version supérieure) ;
- MySQL en version 4.1.2 (ou une version supérieure).

WordPress est non pas un logiciel qui se lance depuis Windows ou OS X, mais un script qui s'installe sur un site web. Il vous faudra donc avant tout disposer d'un hébergement web (gratuit ou payant) remplissant les conditions précédentes et d'un logiciel de transfert de fichier (client FTP), par exemple FileZilla pour Windows ou Transmit pour OS X. Tout est expliqué au Chapitre 2.

Avec ce livre, nous souhaitons vous faire entrer de plain-pied dans la communauté des utilisateurs aguerris de WordPress. Une communauté vieille de sept ans déjà, et dont l'histoire remonte à plus loin encore, jalonnée d'événements marquants qui l'ont façonnée telle qu'elle est désormais. En vous plongeant dans son histoire, vous comprendrez mieux les choix qui ont fait de WordPress le logiciel qu'il est aujourd'hui.

De b2 à WordPress

WordPress trouve ses origines en Corse, île de naissance du développeur Michel "michel_v" Valdrighi, qui le 12 juin 2001 se lança dans le développement de son propre logiciel de publication de blog. Ce logiciel, baptisé b2 en référence à Blogger (qui à l'époque fonctionnait sporadiquement), fait le pari de se reposer sur une base de données (MySQL) plutôt que sur des fichiers statiques, afin d'éviter le laborieux processus de régénération des pages HTML à chaque publication – processus sur lequel reposait Movable Type à ses débuts.

Le développement de b2 est open-source et se fait donc publiquement par le biais du blog de test hébergé sur cafelog.com (ce qui explique l'appellation de "b2/cafelog" fréquemment donnée à ce logiciel), introduisant le concept de marqueurs de modèle dès le 16 juin (`the_content()`, `the_title()`, `the_author()`…), système qui est toujours en vigueur dans WordPress neuf ans après. Plongé dans le développement, michel_v ajoutait de nombreuses fonctionnalités, jusqu'à ce qu'il disparaisse de la surface d'Internet en novembre 2002 (date de la dernière version de b2, la 0.6.1) pour des raisons personnelles.

michel_v étant le seul développeur du projet, son absence prolongée a commencé à inquiéter sérieusement les utilisateurs de b2, qui ont été jusqu'à lancer des appels à information. Parmi ces développeurs inquiets, certains prirent les choses en main pour corriger les problèmes découverts dans b2. C'est là l'une des grandes forces du mouvement open-source : si un projet est abandonné par son auteur, n'importe qui peut reprendre le flambeau. Ainsi, François Planque se mit à retoucher le code pour son propre usage, en avril 2003, ce qui l'amena à sortir publiquement sa version, b2evolution, dès le mois de mai 2003.

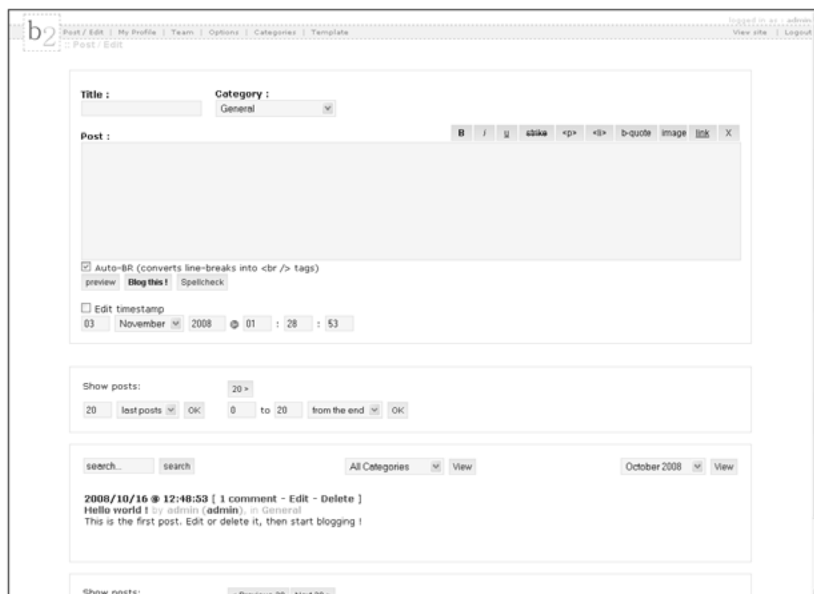


Figure 1.03
Interface d'administration de b2 0.6.2.2.



Figure 1.04
Thème par défaut de b2.

De son côté, Matt Mullenweg lançait dès janvier 2003 un appel à suggestions, auquel répondit aussitôt Mike Little. Tous deux se lancèrent dans la correction des bogues et l'ajout de fonctionnalités. L'idée initiale de Matt était de profiter de la licence open-source de b2 pour en faire un logiciel combinant "la flexibilité de Movable Type, la sémantique de TextPattern, l'extensibilité de b2 et la simplicité d'installation de Blogger". Le logiciel GreyMatter, de Noah Grey, tient également une place importante dans les inspirations de WordPress.

Leur évolution de b2, baptisée WordPress par une amie de Matt, sortit le 27 mai 2003, en version 0.70. Cette première version de WordPress suivait de quelques jours l'annonce

par Michel lui-même, revenu à la surface, que WordPress devenait le successeur officiel de b2. Michel allait d'ailleurs rejoindre l'équipe de développement peu après. Le code de WordPress tout comme son interface d'administration et son thème par défaut seront grandement revus et stabilisés avec la version 0.72 (voir Figures 1.05 et 1.06).

Figure 1.05

Administration de WordPress 0.72.

Figure 1.06

Thème par défaut, créé par Dave Shea pour la version 0.72, aujourd'hui nommé Classic.



Pour autant, le projet b2evolution continue d'être développé à ce jour, en ayant son petit succès.

Évolutions de WordPress de 2003 à aujourd'hui

2003-2004 : des origines à la reconnaissance

Comme il se doit, le projet WordPress a bien évolué depuis cette première version, dans de nombreux domaines, et il serait impossible de lister l'ensemble de ces évolutions. Nous n'en donnerons qu'un aperçu ici.

L'année 2003 a été consacrée principalement à consolider le code, à corriger de nombreux bogues, ainsi qu'à introduire de nouvelles fonctionnalités. Toutes ces modifications ont donné la version 1.0 le 3 janvier 2004, très rapidement suivie par la version 1.0.1, qui corrigeait de nombreux bogues et fut baptisée "Miles" en hommage à Miles Davis.



Figure 1.07
Administration de WordPress 1.0.1.

C'est avec la version 1.0.1 (voir Figure 1.07) que l'habitude a été prise de baptiser les versions de WordPress en hommage aux grands musiciens de jazz, musique dont Matt Mullenweg est friand, lui-même étant un joueur de saxophone. Cet usage a été appliqué à la version mineure suivante (la version 1.0.2 de mars 2004, baptisée "Blakey" pour Arthur Blakey), puis a été réservé aux seules versions majeures.

C'est d'ailleurs avec la version majeure suivante, WordPress 1.2 "Mingus", diffusé le 22 mai 2004, que le projet a connu de grands bouleversements (voir Figure 1.08). Tout d'abord, cette version présentait pour la première fois des avancées notables sur la concurrence du moment : architecture d'extension (auparavant, l'extensibilité de WordPress reposait sur un fichier tiers, wp-hacks.php), internationalisation (les premières équipes de traduction s'étaient formées dans les pays non anglophones), sous-catégories, outil intégré de lutte contre le spam (un système de liste noire), etc.

Figure 1.08
Administration
de WordPress 1.2.

Mais, surtout, une lutte politique entoure cette version. En effet, une dizaine de jours avant la sortie de WordPress 1.2, la société Six Apart annonça une nouvelle politique de prix pour son produit Movable Type, ainsi qu'un changement de licence d'utilisation. Auparavant, Movable Type était simplement gratuit pour un usage personnel, et payant pour ceux qui voulaient vendre des services basés sur le logiciel. La nouvelle situation rendait l'utilisation de Movable Type gratuite pour un auteur et trois sites au maximum, payant au-delà, à partir de 100 dollars. Un très grand nombre de blogueurs impliqués ont donc préféré quitter Movable Type pour WordPress ; notamment, le blogueur et développeur Mark Pilgrim a préféré faire don à WordPress des 535 dollars que lui aurait coûté une licence Movable Type. La version 1.2 marque donc un tournant majeur pour le logiciel, du fait que de nombreux blogueurs de premier ordre ont commencé à utiliser WordPress à ce moment-là : le projet prenait clairement position parmi les principaux outils de blogging.

Il reste que Movable Type permettait de gérer plusieurs blogs distincts à partir d'une même installation, ce que WordPress ne proposait pas. Plusieurs variations de WordPress et b2 proposaient cependant cette capacité multiblog : b2evolution, Lyceum, ou encore b2++. Ce dernier était géré par l'Irlandais Donncha O'Caoimh, qui durant l'été 2004 a rejoint l'équipe WordPress pour créer une version multiblog du logiciel à partir du code de b2++. WordPress MU 0.1 sortira dès octobre 2004 et prendra de plus en plus d'importance, tout en restant un projet distinct de WordPress. Ainsi, WPMU profite toujours des innovations de WordPress, mais avec quelques semaines de retard ; les innovations propres à WPMU ne sont jamais apportées à WordPress.

2005-2008 : la stabilisation et la professionnalisation

La version majeure suivante de WordPress est la 1.5, sortie en février 2005, baptisée "Strayhorn" en référence au pianiste Billy Strayhorn, et qui introduit le fameux thème Kubrick créé par Michael Heilemann et utilisé par défaut pour toute nouvelle installation de WordPress (voir Figure 1.09) – l'ancien thème par défaut devenant le thème Classic. Kubrick a été créé pour exploiter les nouvelles possibilités de l'API de création de thèmes, entièrement remaniée, à l'instar de l'API de création d'extensions.

Figure 1.09
Nouveau thème par défaut, baptisé Kubrick.



WordPress 1.5 introduit par ailleurs le concept de "pages", qui sont des entrées en dehors de la chronologie des articles du blog, et qui font passer WordPress du statut de moteur de blog à celui plus complet de CMS, appellation que les développeurs de WP s'appliqueront à mériter avec les évolutions suivantes (voir Figure 1.10).



Figure 1.10
Administration de WordPress 1.5.

Cependant, ce remaniement du fonctionnement interne de WordPress apportera son lot de problèmes de sécurité, avec pas moins de quatre versions mineures en deux mois (la fameuse série 1.5.1, 1.5.1.1, 1.5.1.2 et 1.5.1.3). WordPress gardera de cette période une réputation de logiciel aux mises à jour trop nombreuses et rapprochées.

En même temps que sortait WordPress 1.5, le projet Shuttle démarrait, dont l'objectif était d'accorder à l'interface d'administration la même qualité graphique que le thème Kubrick – le créateur de ce dernier faisant d'ailleurs partie de l'équipe. Le projet Shuttle était mené par des indépendants habitués de WordPress, mais ne faisant pas partie de l'équipe principale de développement. Une fois leur projet terminé, sur la base de WordPress 1.5, les modifications apportées au code de la version majeure à venir de WordPress étaient telles que seuls quelques éléments furent incorporés ici et là. Dépités, les développeurs en charge de Shuttle se lancèrent dans le développement d'Habari, un logiciel de blogging dont l'objectif était d'utiliser les dernières technologies (PHP 5, MySQL 5, etc.).

Cette version majeure était WordPress 2.0 "Duke" (pour Duke Ellington), sortie en décembre 2005 après de longs mois d'attente. En plus d'une zone d'administration entièrement remaniée (voir Figure 1.11), en partie grâce au projet Shuttle, WordPress 2.0 offrait de vastes améliorations et modifications, justifiant ce saut de version (à l'origine ce devait être la version 1.6). Entre autres choses, le blogueur disposait désormais d'une interface d'écriture dite "WYSIWYG" basée sur TinyMCE, d'une installation par défaut de la célèbre extension antispam Akismet, d'un gestionnaire de rôles pour les utilisateurs, d'un système interne de cache…

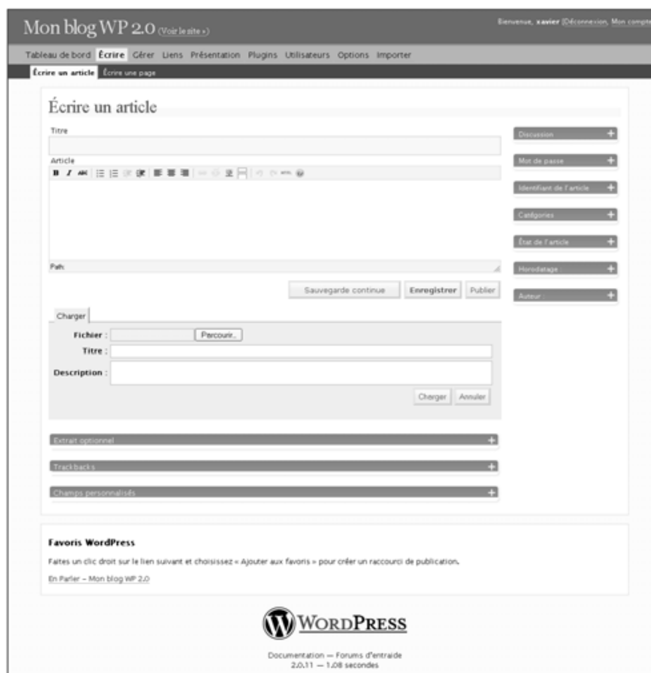


Figure 1.11
Administration de WordPress 2.0.

Les longs mois passés entre les versions 1.5 et 2.0 n'ont cependant pas eu comme seul résultat qu'une nouvelle version majeure de WordPress, aussi importante fût-elle. En août 2005, Matt Mullenweg, le principal responsable de WordPress, créa Automattic, une structure officielle pour héberger les divers projets qu'il chapeautait, et qui deviendra une société à part entière peu avant la sortie de WP 2.0.

C'est que les projets menés par Matt sont nombreux : outre WordPress et le service antispam Akismet, Automattic héberge également le projet bbPress, un outil simple de gestion de forum créé fin 2004, et surtout le service WordPress.com, mis en ligne en août 2005 et ouvert à tous en novembre de la même année. WordPress.com est une version hébergée de WordPress, ou plutôt de WordPress MU, qui a fait bien du chemin depuis son lancement moins de dix mois plus tôt.

Tous les projets menés par Automattic sont disponibles en open-source, mis à part Akismet pour des raisons compréhensibles de protection des algorithmes utilisés…

Suivront régulièrement des mises à jour de sécurité et de correction de bogues pour la branche 2.0.x, tandis que l'équipe travaillait à la prochaine version majeure. Parvenue à la version 2.0.9, elle est considérée comme très stable. Dans les faits, c'est la dernière version de WordPress à fonctionner avec MySQL 3.23. L'équipe de développement, qui avait annoncé vouloir maintenir cette version jusqu'en 2010 (soit 5 ans), a finalement préféré abandonner cette branche, depuis longtemps obsolète.

Janvier 2007 a vu la sortie d'une nouvelle version majeure, la 2.1, baptisée "Ella" pour Ella Fitzgerald. Celle-ci introduit quelques fonctionnalités intéressantes comme la sauvegarde automatique, un nouveau format interne d'import-export et un vérificateur d'orthographe. Mais avec la 2.1, l'équipe prend la décision de se baser sur un nouveau cycle de développement, proche de celui du système Ubuntu Linux : faire au moins une version majeure tous les quatre mois, avec de possibles versions mineures ici et là.

Suivront ainsi, en mai 2007, WordPress 2.2 "Getz" (pour Stan Getz) et la 2.3 "Dexter" en septembre 2007 (pour Dexter Gordon). Ces deux versions apportent comme on peut l'attendre des innovations certaines : support des widgets, support Atom complet, options de configuration avancée (`DB_CHARSET`, `DB_COLLATE`, `WP_SITEURL` et `WP_HOME`), système de gestion de mots-clefs (tags), notification de nouvelle version, révision graphique de l'éditeur de texte, système de taxinomie interne, etc.

2008 : la quête de l'interface idéale

La version 2.4 était normalement prévue pour décembre 2007, puis janvier 2008, du fait du grand nombre de modifications et des vacances des développeurs. Finalement, la version 2.4 fit place à la version 2.5 "Brecker" (pour Michael Brecker), sortie en mars 2008, six mois après la version 2.3. Le cycle de développement a donc évolué pour prendre en compte les ralentissements dus aux congés d'hiver : l'équipe en restera désormais à trois versions majeures par an.

Six mois entre deux versions majeures, mais les différences sont notables. La plus importante est le tout nouvel aspect de l'interface d'administration, créée en collaboration avec les designers de Happy Cog (dont le fameux Jeffrey Zeldman, gourou du design web) [voir Figure 1.12]. Cette interface cherche à mettre en avant les principales fonctionnalités de publication, au risque de déboussoler certains utilisateurs habitués à la précédente interface, en place depuis de nombreuses années.

Mais la version 2.5 n'est pas qu'une nouvelle interface : elle profite d'un tout nouveau système de gestion de médias (images, sons, vidéos) et de création de galeries, d'une notification de mise à jour pour les extensions ainsi que d'une API de "shortcode", d'une forte emphase donnée à la sécurisation de la connexion, et bien d'autres ajouts et corrections.

WordPress 2.6 "Tyner" (pour McCoy Tyner) est sorti de son côté un mois avant la date prévue, mi-juillet 2008, et est construit sur les avancées de la version 2.5 : parcours de l'historique d'un article (comme dans un wiki), implémentation de Gears pour accélérer l'administration, prévisualisation du thème choisi, révision du gestionnaire d'image, support SSL…

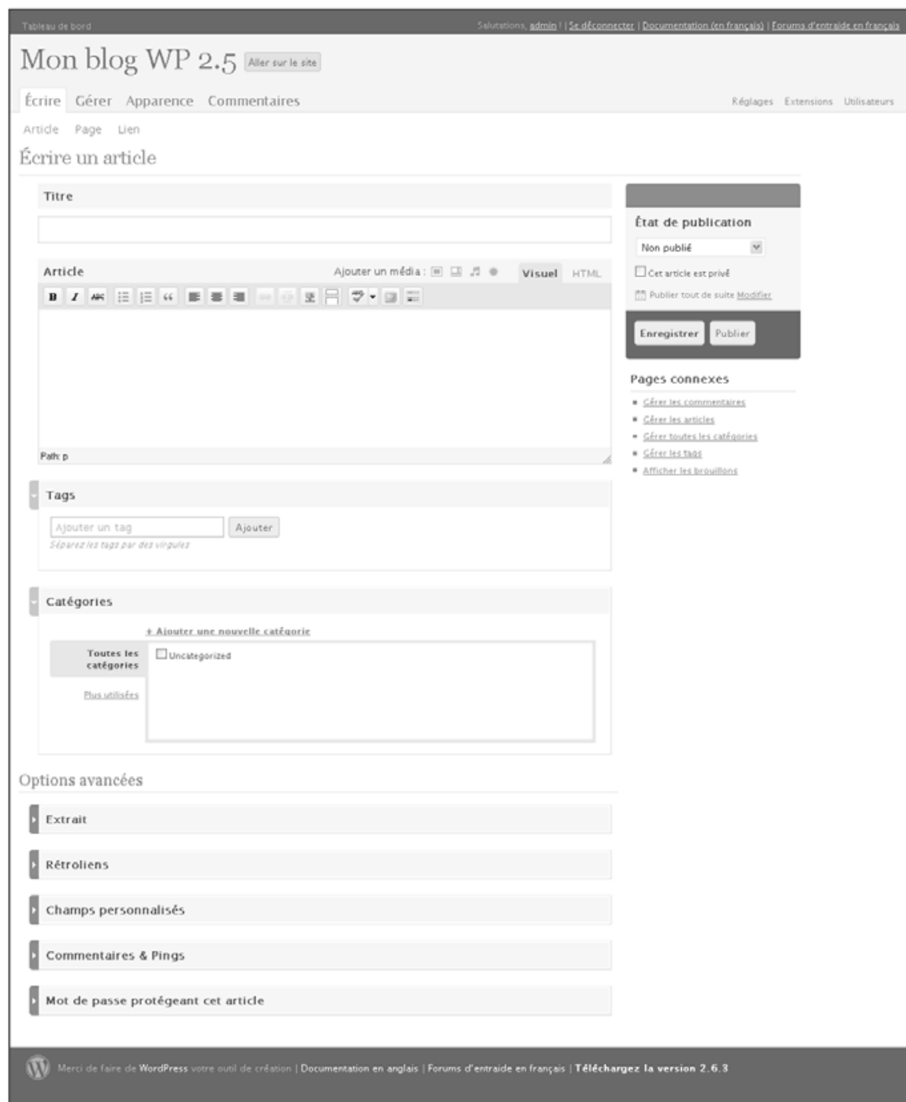


Figure 1.12
Administration de WordPress 2.5.

L'aventure continue avec WordPress 2.7 "Coltrane" (pour John Coltrane), diffusé en décembre 2008, et qui est une nouvelle révolution, de la taille de la version 2.5. En effet, la nouvelle interface de cette dernière version n'a pas été si bien reçue par les utilisateurs, et les développeurs ont donc cherché à voir si en repartant de zéro ils pouvaient créer une meilleure interface, plus accessible et agréable à l'usage (voir Figure 1.13). Conçue sous le nom de code CrazyHorse, cette nouvelle interface déplace le menu principal de WordPress depuis le haut vers la gauche, dans une colonne. Pour bien faire, Automattic a financé des tests oculométriques pour comparer les deux versions. Ces derniers ont révélé que CrazyHorse était plus efficace, aussi cette nouvelle interface a été intégrée à WordPress.

Figure 1.13
Administration de WordPress 2.7, issue du projet CrazyHorse.

Mais WordPress 2.7 va également très loin dans les fonctionnalités : mise à jour automatique de WordPress, des thèmes et des extensions ; intégration d'un moteur de recherche et d'installation automatique d'extensions et de thèmes ; nouvelle API pour gérer ses commentaires en dehors de WordPress ; commentaires hiérarchisés et paginés ; mise en avant d'article (Sticky Post)… L'objectif avec cette nouvelle version majeure est véritablement de simplifier la vie du blogueur et de lui éviter de devoir toucher directement aux fichiers de WordPress.

2009 : préparer l'évolution vers de nouveaux sommets

Disposant désormais d'une base solide avec la 2.7, l'équipe de développement travaille à améliorer toujours plus la vie des utilisateurs comme des développeurs. Cela passe, en partie, par une adaptation de la nouvelle interface graphique aux besoins de chacun, et nombre

de questions relatives à l'interface sont désormais directement posées à la communauté par le biais de sondages.

En avril 2009, le projet BuddyPress arrive enfin à maturité avec sa version 1.0, après plus d'un an de développement pour Andy Peatling, son seul développeur, entre-temps embauché par Automattic.

WordPress 2.8 "Baker" (pour Chet Baker) est lancé en juin 2009, après donc plus de cinq mois de développement. On pourrait s'attendre, après tant de temps, à des avancées spectaculaires, mais les développeurs ont surtout choisi de consolider l'existant, de rendre WordPress plus rapide et léger et de préparer l'avenir. Pour autant, les apports sont notables : installation et mise à jour automatique des thèmes, comme il était déjà possible de le faire pour les extensions dans la 2.7 ; refonte complète de la page de gestion des widgets, ainsi qu'une nouvelle API de widgets ; et, surtout, des centaines de petites améliorations, tant pour les utilisateurs que pour les développeurs.

Bien que la 2.8 fût une version ayant eu un grand nombre de contributeurs, elle aura quand même six mises à jour mineures (jusqu'à la 2.8.6), et notamment trois en un mois, pour des raisons de sécurité, ce qui relance les questions sur la protection offerte par WordPress... Du fait de sa grande popularité, le projet WordPress est régulièrement la cible de personnes malveillantes, et l'instauration d'un outil de mise à jour automatique permet aux utilisateurs de se prémunir plus rapidement et simplement d'attaques potentielles.

L'année 2009 est également marquée par un réel succès des thèmes payants, et par un débat intense dans la communauté sur le fait de devoir diffuser ses thèmes sous licence GPL (donc, gratuitement). L'équipe de développement, partisane du tout-GPL, a été confirmée dans son choix par le Software Freedom Law Center. Nombre de développeurs ayant réussi ont décidé de suivre cet enseignement et de passer au tout-GPL. Pour autant, le sujet reste encore l'objet de nombreux débats enflammés...

2009 marque aussi l'abandon de la branche 2.0.x de WordPress. Celle-ci devait être maintenue jusqu'en 2010 afin de respecter la norme du projet Debian, mais les mises à jour n'étaient que sporadiques, et les problèmes de sécurité ne faisaient que rendre cette branche plus dangereuse.

C'est en décembre que sort enfin la version 2.9, plus de six mois après la précédente version majeure, secouée par de nombreuses mises à jour de sécurité. Baptisée "Carmen" en l'honneur de la chanteuse de jazz Carmen McRae, elle introduit principalement quatre nouveautés : un éditeur d'images intégré (voir Figure 1.14), afin de couper, pivoter ou redimensionner les images depuis l'interface de WordPress ; un système de Corbeille pour récupérer les contenus effacés trop rapidement ; un système communautaire d'évaluation de la compatibilité des extensions, afin de s'assurer qu'elles fonctionnent correctement et de minimiser les risque de "cassure" du blog ; enfin, une prise en compte facilitée des contenus en provenance d'autres sites, notamment les vidéos (YouTube, DailyMotion, etc.). Et, comme d'habitude, de nombreux autres ajouts et corrections moins visibles, notamment à destination des développeurs et créateurs de thèmes.



Figure 1.14
L'éditeur d'image intégré, en action.

2010 : la grande intégration

La version 2.9 désormais disponible, tous les regards se portent vers la version 3.0. Malgré le fait que Matt Mullenweg a toujours annoncé que les numéros des versions majeures se suivaient sans lien avec l'importance de chaque version, toute la communauté attend une évolution majeure pour cette version. Leurs attentes sont satisfaites peu de temps avant la sortie de la version 2.8 : dès mai 2009, Matt a commencé à annoncer dans les différents WordCamps auxquels il participait autour du monde que la version 3.0 serait l'occasion d'intégrer totalement le code multiblogs au sein de WordPress et de ne plus avoir qu'une version. À partir de WordPress 3.0, WordPress MU n'aura plus de raison d'être : les fonctionnalités multiblogs seront directement accessibles pour toutes les installations de WordPress – moyennant quelques réglages inaccessibles aux débutants, heureusement...

Pour les projets open-source gérés par Automattic, une étape importante est franchie fin janvier 2010, avec le lancement de la WordPress Fundation, chargée de gérer le code source et les marques déposées liées à ces projets. Automattic n'est donc plus qu'une société qui participe au développement de ces projets.

Autre étape importante avec la 3.0 : l'abandon de Kubrick comme thème par défaut. En place depuis WordPress 1.5 (2005), ce thème a certes permis à WordPress de se faire une place parmi les blogs "beaux dès l'installation", mais est depuis devenu un peu trop envahissant, nombre de blogueurs ne prenant pas le temps de mettre en place un autre thème. Le nouveau thème, baptisé *Twenty-Ten* ("2010" en anglais) permettra par ailleurs d'intégrer toutes les nouvelles API mises en place depuis les dernières versions, et que Kubrick ne reconnaissait pas (voir Figure 1.15).

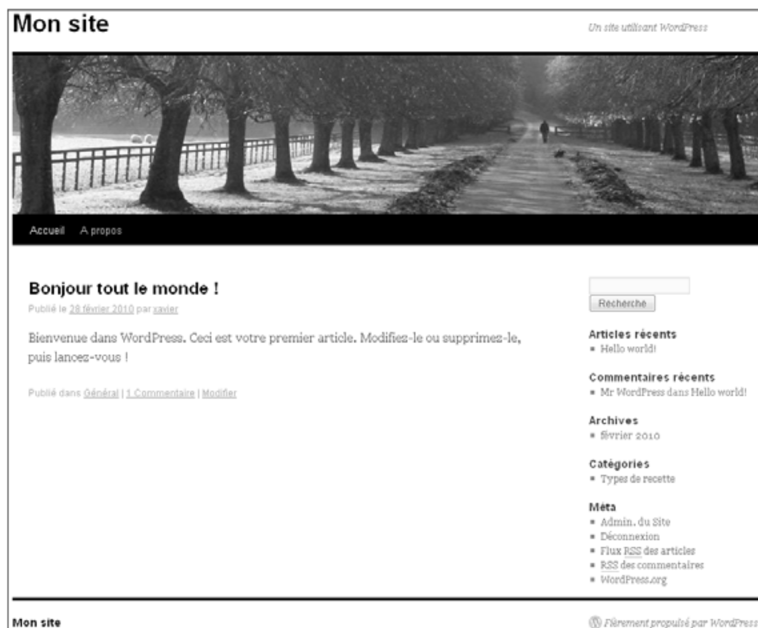


Figure 1.15
Twenty Ten, le nouveau thème par défaut de WordPress.

WordPress 3.0 sort enfin le 17 juin 2010, six mois presque jour pour jour après le lancement de son développement. Outre la fusion avec WPMU et le nouveau thème, cette version présente quelques améliorations de première importance. Parmi celles-ci, l'arrivée d'un gestionnaire de menus qui permet (pour les thèmes qui peuvent l'exploiter) de gérer le menu du thème directement depuis l'administration, sans devoir modifier le code HTML ou PHP ; cet outil, basé largement sur les travaux de l'équipe de WooThemes, facilite énormément le quotidien des utilisateurs. Autres améliorations de la version 3.0, les types de contenus personnalisés et taxinomies personnalisées (*custom post types* et *custom taxonomies*) propulsent WordPress dans le groupe des CMS robustes et complets – au point de mériter un nouveau chapitre pour le présent livre. Par ailleurs, toujours dans un souci de rendre WordPress plus simple d'accès, l'ensemble des pages de l'administration dispose désormais d'une aide contextuelle très complète (et totalement traduite en français...).

L'avenir

La version 3.0 a été un véritable marathon, où l'ensemble des développeurs a dû travailler d'arrache-pied des semaines durant pour l'intégration de certaines fonctionnalités. De fait, il a été décidé de ne pas attaquer immédiatement le développement de la version 3.1, mais de dédier l'été 2010 à l'amélioration des outils communautaires, à commencer par la refonte totale du site WordPress.org. Ce projet est baptisé "3.org", et devrait voir non seulement le site officiel transformé en réseau social grâce à BuddyPress, mais aussi une refonte du Codex, des dépôts de thèmes et d'extensions, des forums, etc. En parallèle, les développeurs prendront le temps d'observer ce que les développeurs et créateurs de thèmes peuvent imaginer avec les nombreuses API de la version 3.0, afin de les améliorer encore pour la 3.1.

La reprise des développements en vue de cette version 3.1 est quant a elle prévue pour début septembre 2010...

Combien vous coûtera WordPress ?

Le fait de dire que WordPress est disponible sous licence open-source n'est pas toujours clair pour tous les utilisateurs – et encore moins quand on mentionne la licence GPL. La présente section est donc nécessaire.

WordPress, parce qu'il est placé sous la licence open-source GPL, est un logiciel gratuit et libre d'usage. Cela signifie donc, tout simplement, les points suivants :

- Vous n'aurez jamais à payer pour l'utiliser.
- Vous pouvez parfaitement vous en servir pour gérer un site web à vocation commerciale – ou pour afficher des publicités, type Google Ads.

Mais, attention, notez bien que seul le logiciel lui-même est gratuit et libre d'usage. Il existe autour de WordPress, comme pour tout logiciel open-source, un véritable écosystème construit au fur et à mesure des besoins des utilisateurs : designers de thèmes, développeurs d'extensions, et autres services. Si de nombreux membres de cette communauté participent à titre gratuit, une portion non négligeable vend ses services – et ils en ont parfaitement le droit. Dans votre quête d'un blog qui vous ressemble vraiment, vous rencontrerez donc certainement des thèmes payants (et même quelques extensions), et nombre de services se proposant d'installer un blog pour vous. Certains vivent très bien de cette activité à plein temps (surtout les designers de thèmes), et nous espérons qu'avec ce livre vous pourrez vous aussi devenir un de ces *lucky few* qui vivent à 100 % grâce à WordPress.

En définitive, utiliser WordPress ne vous coûtera que le temps que vous voudrez bien passer à (ou apprendre à) vous en servir, et en retour vous offrira une forme de liberté : plutôt qu'utiliser un outil fermé (payant ou non) d'où vous ne pouvez pas partir sans lui abandonner vos données, WordPress peut être installé chez tous les hébergeurs et dispose d'outils d'import-export vous assurant de toujours pouvoir faire héberger votre site là où il vous sied. Cette liberté des données est une spécificité non pas des logiciels open-source mais des formats ouverts qui en découlent... Mais c'est un autre débat.

Bien sûr, il vous faudra quand même payer votre hébergeur, ainsi que votre fournisseur d'accès à Internet !

À qui appartient WordPress ?

La réponse standard est : WordPress n'appartient à personne, puisque le code du projet est sous licence open-source.

C'est l'un des grands principes de l'Open Source : le code n'appartient à personne, n'importe qui peut créer une version alternative de WordPress (un *fork*) et en faire ce que bon lui semble, tout dépend de la communauté qu'il parviendra à réunir autour de sa version. C'est ce qui s'est passé quand Matt Mullenweg et Mike Little ont décidé de reprendre b2/cafelog pour en faire WordPress...

Cependant, la présence de la société Automattic tend à faire douter les plus sceptiques. Créée par Matt Mullenweg, gestionnaire de WordPress.com, propriétaire des serveurs qui

hébergent le code WordPress, employeuse des principaux développeurs de WordPress... Nombre de faits portent à croire que, malgré la licence open-source, Automattic garde la mainmise sur le code et pourrait en faire ce que bon lui semble.

C'est une fois de plus oublier que la licence autorise n'importe quel mécontent à prendre l'intégralité du code et à aller le faire développer ailleurs, avec la communauté idoine. Plus prosaïquement, si Automattic emploie effectivement nombre de personnes qui travaillent exclusivement sur WordPress, pas moins de quatre *core developers* du projet ne doivent pas leur pain quotidien à cette société, soit plus de la moitié.

Reste que tout porte à croire que c'est Automattic qui tire les ficelles ; aussi, pour parer à ce type de critique, Matt Mullenweg a lancé en 2010 un projet qui lui tenait à cœur : la Fondation WordPress. Celle-ci est aujourd'hui propriétaire du code ainsi que des marques liées à WordPress, BuddyPress, bbPress et le concept de WordCamp. Il s'agit d'une association à but non lucratif ("501(c)3 non-profit organization"). Automattic et Matt Mullenweg ne sont donc pas propriétaires de WordPress... du moins directement.

La communauté francophone de WordPress

Les francophones ont longtemps dû utiliser WordPress en anglais et n'avaient que les forums officiels (anglophones) où poser leurs questions. De fait, WordPress n'a été accessible pendant longtemps qu'aux personnes bilingues.

WordPress n'a pu être correctement traduit qu'à partir de la version 1.2 (mai 2004), qui ajoutait quelques fonctionnalités allant dans ce sens, et surtout la version 1.5 (février 2005). Effort collectif à l'origine réalisé par le biais d'un wiki, la traduction a évolué rapidement grâce aux outils normalisés reconnus par WordPress 1.5. C'est avec cette version que la traduction a été entièrement remaniée collectivement, par le biais du logiciel en ligne Rosetta. Enfin, sous un format stable (le standard gettext, utilisant des fichiers .po/.mo), l'équipe de développement a laissé la maintenance de la traduction à des volontaires, Myriam Faulkner et Xavier Borderie, rejoints par la suite par Sébastien Erard, puis Amaury Balmer.

Le besoin de forums d'entraide en français se faisant cependant de plus en plus sentir, le site communautaire WordPress Francophone est lancé en août 2005 par une poignée d'utilisateurs : Matthieu Bellon, Damien Gayrard et Xavier Borderie. Cette équipe de bénévoles s'agrandit rapidement avec l'arrivée de volontaires enthousiastes : Amaury Balmer, Benoît Catherineau et Hubert Selliah, puis Arnaud Mangasaryan et Thomas Grim. L'adresse du site est **http://wordpress-fr.net/** (voir Figure 1.16).

WordPress Francophone s'occupe également d'organiser, chaque année depuis 2008, l'édition parisienne du WordCamp (**http://wordcamp.fr**), sous la forme d'un barcamp où chacun vient partager ses connaissances et rencontrer les autres participants de la communauté. L'édition 2009 a d'ailleurs reçu la visite de Matt Mullenweg.

Figure 1.16
Page d'accueil du site WordPress Francophone.



La communauté orbite principalement autour du forum géré par les utilisateurs eux-mêmes, avec l'assistance d'une poignée de modérateurs, ainsi que la traduction de WordPress.

La communauté francophone ne se limite heureusement pas au seul site WordPress Francophone : nombreux sont les utilisateurs ou créateurs de thèmes et extensions à lancer de leur côté des initiatives personnelles pour assurer un support plus particulier. Parmi les plus connus, vous trouverez, par ordre alphabétique :

- Amaury Balmer, créateur d'extensions et de thèmes (**http://www.herewithme.fr/**), traducteur de WordPress et de bbPress (**http://bbpress.fr/**) ;
- Bruno Bichet, actualités et astuces (**http://www.css4design.com/**) ;
- Francis Chouquet, créateur de thèmes (**http://fran6art.com/**) ;
- Maxime Guernion, actualités et astuces (**http://blogtoolbox.fr/**) ;
- Henri Labarre, actualités et astuces (**http://www.2803.fr/**) ;
- Yann L'Hostis, traducteur de thèmes et d'articles (**http://wordpress-tuto.fr/**).

Terminologie de la blogosphère

Comme tout corps de métier, le Web dispose de son vocabulaire propre, connu aujourd'hui de la plupart des internautes, même débutants : navigateur, site, e-mail, pop-up, spam, foire aux questions, smiley… Le monde des blogs lui-même dispose d'un vocabulaire très spécifique, auquel il faut s'habituer si l'on souhaite par exemple comprendre les discussions relatives aux flux RSS, aux podcasts ou aux rétroliens.

La plupart de ces mots sont utilisés tout au long de cet ouvrage, il est donc dans votre intérêt de les connaître afin de savoir de quoi il retourne dans les pages qui suivent.

En raison de l'origine souvent anglophone des technologies du Web, nombreux sont les néologismes, voire les emprunts directs. Ici nous mettons en avant les termes qui semblent les plus couramment admis ; vous trouverez néanmoins des traductions et des synonymes après leur définition.

Vocabulaire du blog

- **Agrégateur.** Outil permettant de recevoir les mises à jour et de lire les derniers articles de blogs choisis par le biais de l'abonnement à leurs flux. Également : lecteur RSS, lecteur de flux, lecteur de news.
- **Archives.** Regroupement d'un ensemble d'articles d'un blog. Le regroupement est le plus souvent chronologique, mais peut également être thématique : par sujet, par catégorie, par auteur…
- **Article.** Élément atomique du blog, comprenant un titre, un contenu, une date et un permalien. Un blog est composé d'une suite d'articles triés par date. C'est la base du blog : sans article, pas de blog. Également : billet, entrée, note ; post, blogpost, entry.
- **Blog.** Site personnel ou d'entreprise, prenant la forme d'une suite d'articles affichés antéchronologiquement (du plus récent au plus ancien). Également : blogue, weblogue, carnet web, cybercarnet, joueb ; weblog, webdiary.
- **Blogiciel.** CMS spécialisé dans la création de blogs. Également : carneticiel ; blogware.
- **Blogosphère.** Ensemble des blogs sur Internet, qui peut être subdivisé en autant de communautés au besoin (francoblogosphère, tricoblogosphère). Également : blogobulle, carnetosphère.
- **Bloguer.** Action de publier sur son blog. Également : carnétiser.
- **Blogueur.** La personne qui écrit les articles du blog. Ils peuvent être plusieurs à écrire pour un seul blog. Également : carnetier, diariste ; blogger.
- **Catégorie.** Regroupement d'articles créé par le blogueur afin de réunir les entrées portant sur le même thème. Un article peut se trouver dans plusieurs catégories. Les catégories forment une méthode hiérarchique de classement des données.
- **CMS.** Outils de gestion de contenu, de l'anglais *Content Management System*, qui simplifient grandement la mise en ligne de contenu multimédia (textes, images, sons, vidéos). Les outils de blog comme WordPress sont des CMS spécialisés dans la gestion de blogs.
- **Commentaire.** Entrée attachée à un article précis, écrite par un lecteur ou le blogueur lui-même, en réponse à l'article ou à un commentaire précédent. Les commentaires forment la base des discussions sur un blog et servent à l'enrichir par les discussions ainsi

engagées. Pour autant, certains blogueurs préfèrent fermer leurs commentaires, ou du moins les ouvrir que pour certains articles.

- **Entrée.** Contenu ajouté par le blogueur. Il peut s'agir d'un article, d'une page, d'un lien, d'un commentaire… Également : item.
- **Flux de syndication.** Fichier contenant les derniers articles, automatiquement mis à jour par le CMS lors d'une nouvelle publication. Ce fichier, difficilement lisible tel quel car basé sur le format XML, est destiné à être lu par les agrégateurs. Il contient l'essentiel de chaque article : titre, date de publication, adresse directe, et au choix un extrait de l'article ou le texte complet. Les deux formats les plus répandus sont les flux RSS (*Rich Site Summary* ou *Really Simple Syndication*) et les flux Atom. Également : fil ; feed.
- **Horodatage.** La date et l'heure de publication d'un article.
- **Metablog.** Blog collectif, collaboratif ou communautaire, pouvant aborder des sujets très éclectiques en fonction de ses membres. Également : métacarnet, métablogue.
- **Métadonnée.** Information associée à une entrée, qui sert à la décrire et qui peut être exploitée par le CMS : auteur, date de publication, catégorie… Également : metadata.
- **Moblog.** Blog écrit principalement par le biais d'un téléphone portable (envoi de photos, de vidéos). Également : moblogue, blogue mobile, mobicarnet.
- **Mots-clefs.** Regroupement d'articles créé par le blogueur afin de réunir les entrées listant les mêmes mots-clefs. Un article peut disposer d'autant de labels que le blogueur l'estime utile. Les mots-clefs forment une méthode non hiérarchique de classement des données et ne doivent donc pas être utilisés comme des catégories. Également : labels ; tags.
- **Permalien.** Lien permanent vers un article, une catégorie, un label ou une date du blog. Permet depuis un flux d'accéder directement à un article, même ancien, sans devoir passer par la page d'accueil du site, ni son moteur de recherche. Sert d'identifiant visuel dans le cas où le blog dispose d'URL propres.
- **Photoblog.** Blog dont les articles contiennent principalement des photographies prises par le blogueur.
- **Ping.** Fonction interne du logiciel de blog, destinée à indiquer à un site lié qu'on parle de lui dans l'article publié. Sert également à signaler une mise à jour aux moteurs d'indexage.
- **Podcast.** Série d'émissions enregistrées au format audio ou vidéo, diffusées par le biais d'un blog. Le lecteur de podcast télécharge le fichier MP3 ou vidéo à partir de son URL contenue dans le flux RSS, sous le nom d'enclosure.
- **Publier.** L'action de mettre un article en ligne, accessible à tous. Un article peut également être prépublié si sa date de publication est dans le futur. Également : poster.

- **Référent.** Sites web ou blogs d'où proviennent les lecteurs. Vérifier ses référents permet de savoir quels sont les sites/blogs qui font un lien vers son blog, ou de connaître les mots-clefs qui affichent son blog dans les moteurs de recherche. Également : referer.
- **Rétrolien.** Lien explicite placé par le blogueur vers un autre blog, afin de lui signaler qu'il est mentionné dans l'article publié. Fait le plus souvent office de doublon avec le système automatisé de ping. Également : pisteur ; trackback.

Quelques technologies et logiciels utiles

- **Apache.** Serveur web. C'est le logiciel avec lequel communique le navigateur et qui se charge d'envoyer les fichiers (code HTML, image, sons, etc.) attachés à une adresse web. Un fichier .htaccess placé à la racine du site permet de contrôler certains aspects du serveur. Gratuit, le serveur Apache est utilisé par la grande majorité des sites web.
- **CSS.** Langage de description permettant de séparer le contenu (texte, image, son) du contenant (mise en page).
- **FTP.** Protocole de transfert de fichiers. Par le biais d'un client FTP, le blogueur peut accéder à son espace web et ainsi mettre en ligne ou effacer des fichiers. C'est la première étape de l'installation d'un blog ou de la mise en place d'un thème ou d'une extension.
- **HTML/XHTML.** Langage de balisage de données permettant de décrire le contenu d'une page web. C'est le socle d'une page web, à partir duquel sont appelés les images, les sons et autres fichiers tiers. Il est surtout utile de le connaître si l'on veut maîtriser le code de ses articles ou créer/modifier son thème.
- **JavaScript.** Langage de programmation web, côté client, permettant de mettre en place une certaine interactivité sur une page web.
- **MySQL.** Système de gestion de base de données (SGBD). Le contenu d'un blog n'est pas stocké dans des fichiers texte, mais sur un serveur MySQL. Gratuit, c'est l'un des SGBD les plus populaires sur le Web actuellement.
- **PHP.** Langage de programmation web, côté serveur. PHP permet de créer selon les besoins des pages HTML différentes, sans intervention de l'utilisateur. Un très grand nombre de sites sont programmés à l'aide de PHP, parmi lesquels Wikipédia, Facebook, Digg et, donc, WordPress.
- **phpMyAdmin.** Logiciel écrit en PHP permettant de gérer une base de données MySQL par le biais d'une interface web plutôt qu'en ligne de commande.

Vocabulaire propre à WordPress

WordPress dispose également d'un langage spécifique, auquel nous aurons recours tout au long de ce livre. De fait, si certaines définitions vous laissent perplexe, ne vous inquiétez pas, tout va s'éclairer dans les chapitres qui suivent.

Côté utilisateur

- **Administration.** Ensemble des pages permettant de gérer le blog, notamment l'écriture d'articles. Cette section n'est accessible qu'au propriétaire du blog et aux coblogueurs qu'il aura désignés.
- **Barre latérale.** Zone annexe du blog, généralement affichée à côté des articles de la page d'accueil et contenant certaines informations : catégories du blog, calendrier, données en provenance de certaines extensions. Également : sidebar.

- **Champ personnalisé.** Métadonnée créée par l'utilisateur ou une extension plutôt que par WordPress. Également : custom fields.
- **Codex.** Site principal de documentation de WordPress. Le Codex repose sur un système de type wiki, ce qui signifie que la documentation peut être améliorée par n'importe qui.
- **Extension.** Programme conçu par un développeur tiers, et qui étend ou modifie le fonctionnement de WordPress. Une extension typique est un filtre antispam.
- **Modèle.** Fichier dynamique utilisé pour générer le code HTML du blog. Un modèle est l'élément atomique d'un thème. Ses fichiers sont écrits en PHP. Également : patron, gabarit ; template.
- **Page.** Entrée publiée en dehors de la chronologie du blog. Les pages fonctionnent de la même manière que les articles, mais sont traitées différemment par WordPress : elles ne peuvent pas appartenir à une catégorie, mais une page peut contenir plusieurs sous-pages. Pour les différencier de l'expression générale de "page web", on parlera ici de "page statique".
- **Rôle.** Identité que peut prendre un membre du blog. WordPress gère cinq rôles : administrateur, éditeur, auteur, contributeur, abonné. À chaque rôle se rattachent des capacités, c'est-à-dire des droits d'accès à certaines fonctionnalités de l'administration.
- **Thème.** Ensemble de modèles PHP, d'images et de fichiers CSS définissant l'apparence du blog pour les visiteurs.

Côté développeur

- **Action.** Crochet utilisé par WordPress au cours de son fonctionnement interne. Il permet au développeur d'extensions de modifier ce fonctionnement à son gré.
- **API.** De l'anglais *Application Programming Interface*, traduit en "interface de programmation", il s'agit d'un ensemble de fonctions mises à la disposition des développeurs de thèmes ou d'extensions, afin d'exploiter les données du blog.
- **Boucle.** Cœur de l'affichage du blog, la boucle parcourt les entrées du blog selon certains critères établis par le développeur du thème et influencés par les actions du visiteur, afin d'afficher du contenu plus ou moins spécifique (tous les articles du blog, tous les articles d'une catégorie, un seul article, etc.).
- **Champ personnalisé.** Métadonnée rattachée à un article ou une page statique.
- **Crochet.** Fonctionnalité de WordPress à laquelle un développeur d'extensions peut attacher du code. Il y a deux types de crochets : l'action et le filtre, la différence la plus visible étant que le filtre renvoie une valeur, au contraire de l'action. Également : hook.
- **Filtre.** Crochet utilisé par WordPress avant d'écrire une donnée dans la base de données ou de l'afficher à l'écran. Le filtre permet au développeur d'extensions de modifier la donnée avant l'écriture ou l'affichage.

- **Marqueur de modèle.** Fonction PHP utilisée dans un modèle, permettant d'y placer un contenu tiré de la base de données du blog. Typiquement, il s'agit des données liées à une entrée : titre, contenu, horodatage, lien, auteur…
- **Taxinomie.** Un regroupement de contenus. Par défaut, les articles peuvent être groupés à l'aide de catégories et de mots-clefs. Un thème ou une extension peut par ailleurs ajouter autant de taxinomies que nécessaire, à n'importe quel type de contenu, par défaut ou personnalisé. Également : taxonomy, custom taxonomy.
- **Type de contenu.** WordPress dispose par défaut de cinq types de contenus : les articles, les pages, les fichiers joints, les révisions et les menus de navigation. Un thème ou une extension peut par ailleurs ajouter autant de types de contenu que nécessaires : les types de contenu personnalisés. Également : custom post types.

Remerciements

Les auteurs tiennent à remercier…

Patricia Moncorgé, notre éditrice chez Pearson, tout d'abord pour avoir monté le projet, mais surtout pour l'avoir soutenu malgré tous les problèmes survenus pendant sa conception. Sans le soutien constant et la grande compréhension de Patricia, ce livre serait obsolète avant même son arrivée en librairie… Merci également à Amandine Sueur et à toute l'équipe de Pearson Education France ainsi qu'à Hervé Guyader, pour nous avoir aidés à terminer ce projet dans les meilleures conditions. Enfin, un grand merci à notre correcteur, Philippe Gérard : sans son travail de fourmi, nos textes seraient sans doute bien moins digestes.

Michel Valdrighi, le créateur de b2, et Matt Mullenweg, le fondateur du projet WordPress, sans qui ce livre n'aurait tout simplement pas de raison d'être. Ils ont eu par ailleurs la gentillesse d'accepter d'écrire chacun un texte introductif, le préambule et la préface.

Stephanie Booth, membre émérite de la blogosphère francophone, nous a été d'une très grande aide en faisant plus que relire simplement nos chapitres : ses conseils et corrections ont largement contribué à rendre la première édition de ce livre encore meilleure. S'il devait rester des fautes et incohérences, elles ne seraient dues qu'à notre propre inattention.

Amaury

J'ai toujours été convaincu que participer à la communauté d'un logiciel libre était un excellent tremplin professionnel. C'est pourquoi j'ai longtemps navigué entre différents gestionnaires de contenu libres, comme Dotclear, XOOPS ou Typo3, en tentant de m'impliquer par les forums, les documentations, et ainsi trouver une place dans leurs communautés. Cependant, lorsqu'une communauté existe déjà depuis longtemps, il n'est pas facile, voire impossible, de "faire son trou".

Par chance, en août 2005, je suis tombé par le plus grand des hasards sur un article d'appel à participation pour construire ce qui allait devenir le WordPress Francophone. Cet appel, lancé conjointement par Matthieu Bellon et Xavier Borderie, allait changer ma vie ! En plus de contribuer au débat d'idées autour du portail francophone, j'ai pris la responsabilité

d'effectuer la majorité des développements techniques nécessaires aux fonctionnalités de WP-FR.

Cinq ans après, et quelques coups de folie de ma part, je tiens à remercier Xavier et Matthieu pour leur confiance et leur patience – chose qui n'a pas toujours dû être facile.

Grâce à cette aventure, j'ai eu l'occasion d'acquérir une expérience unique et une reconnaissance dans ce domaine. J'ai donc pu créer mon entreprise, WP-Box, et je suis désormais un entrepreneur expert dans les technologies WordPress et BuddyPress.

C'est également grâce à cet investissement bénévole que je me retrouve ici à écrire un livre sur mon sujet de prédilection, WordPress – chose que je n'aurais jamais imaginée il n'y a pas si longtemps que cela...

Pour finir, je tiens également à remercier ma compagne, Meriem, et ma famille, pour m'avoir permis d'arriver sans encombre au bout de la rédaction de ce livre.

Francis

Quand j'ai démarré mon blog en 2006, ce n'était pas dans l'optique de parler de WordPress. Cela s'est fait naturellement, comme souvent sur les blogs. J'ai adoré la plate-forme et surtout sa communauté. J'en ai donc parlé, mes articles ont été lus, et de fil en aiguille, fran6art.com est devenu une solide base de données sur WordPress.

Ensuite, tout s'est enchaîné. J'ai décidé de faire des blogs et de WordPress mon métier, et un jour la possibilité s'est présentée de faire un livre sur ma plate-forme de prédilection. Et tout cela parce qu'un jour j'ai créé un blog et que j'ai décidé de parler de WordPress ! S'il vous fallait un exemple du pouvoir des blogs, ne cherchez plus !

Écrire un livre n'a rien d'évident. Même si le sujet est passionnant, il faut trouver le bon axe pour l'aborder, et trouver les bonnes formules. Lorsque le projet a été évoqué, j'ai tout de suite pensé à Xavier et Amaury pour leur implication dans la communauté francophone, mais aussi pour leurs qualités techniques et humaines. Rédiger un ouvrage tel que celui-ci avec nos différentes visions de l'outil ne pouvait être que bénéfique pour le lecteur. Je tiens à les remercier tout particulièrement pour avoir accepté cette mission. Sans eux, le livre n'aurait pas cette consistance.

Je voudrais aussi remercier tout simplement ma famille, et en premier lieu ma femme, Silja, qui m'a toujours soutenu et qui me donne chaque jour l'opportunité d'accomplir mes rêves. Un grand merci aussi à ma fille, Lola, mon plus grand bonheur, ainsi qu'à mes parents, pour avoir toujours été derrière moi, même quand la route était plus que sinueuse.

Xavier

J'ai commencé à bloguer en février 2003, avant tout pour me forcer à écrire afin de mieux me remémorer les petits moments du quotidien, et donc aider ma mémoire trop souvent défaillante. J'utilisais l'outil pMachine, mais je me suis rapidement intéressé à b2, puis à son

successeur WordPress. J'ai sauté le pas et ai installé WordPress en juin 2004, mais je m'étais impliqué dans sa traduction dès le mois de mars.

C'est en travaillant de manière ouverte avec d'autres utilisateurs que je me suis pris de passion non seulement pour ce logiciel, mais surtout pour le mouvement open-source et sa croyance fondamentale dans le bénéfice des autres – pour moi excellemment résumé par le mot "ubuntu" : "Je suis ce que je suis grâce à ce que nous sommes tous."

De fil en aiguille, je suis devenu le principal responsable de la traduction française, puis le cofondateur de la communauté française, et aujourd'hui le président de l'Association des utilisateurs francophones de WordPress. Après toutes ces années, je reste avant tout un passionné, animé par l'apport de la communauté et l'amour du travail bien fait.

Pour le projet de longue haleine dont le livre que vous tenez entre vos mains représente l'aboutissement, je tiens tout d'abord à remercier mes compagnons de route : Francis pour m'avoir proposé de m'y investir et pour nous avoir rappelé la nécessité de boucler dans les temps ; Amaury pour sa pertinence technique ; Stephanie pour son accompagnement au quotidien lors de la première édition ; et Patricia pour sa volonté de publier un bon livre sur WordPress plutôt qu'un livre de plus sur le sujet…

Écrire un livre technique requiert tout autant d'être un passionné d'informatique qu'un amoureux des lettres. Si j'ai la chance de combiner ces deux intérêts, c'est en premier lieu grâce à mes parents et à mes deux frères, qui ont respecté et encouragé ces passions, et à mes amis, qui ont contribué à faire de moi ce que je suis. J'espère qu'ils sont contents du résultat !

Un salut particulier à Fabrice Le Guernec, qui depuis plusieurs années m'encourage à écrire un livre ; il ne s'attendait probablement pas à ce résultat, mais l'intention est là.

Partie 

WordPress côté utilisateur

2 Installer WordPress

Lorsque vous souhaitez créer un blog en ligne avec WordPress, vous avez deux possibilités :

- **WordPress.com.** Vous n'avez pas envie de vous compliquer la vie avec un hébergement et la maintenance d'un blog en ligne ? Alors, WordPress.com est probablement la solution pour vous. En effet, en cinq minutes, votre blog est en ligne et avec votre adresse web personnalisée. Choisissez votre design parmi les différents thèmes proposés, et il ne vous restera plus alors qu'à rédiger vos articles. WordPress.com est en fait une solution de blogs "autohébergés". Pas besoin d'installer quoi que ce soit, ce sont les hébergeurs qui s'occupent de tout. Vous n'avez qu'à vous concentrer sur le contenu et l'apparence de votre blog.
- **WordPress.org.** C'est le sujet de notre livre. Si les blogs hébergés sont intéressants pour les utilisateurs néophytes, ils ne vous permettent pas de prendre en main votre blog et d'avoir entièrement le contrôle sur ce que vous pouvez en faire. Avoir un blog que vous hébergez vous-même et pour lequel vous avez tous les accès vous permettra de créer une structure personnalisée et paramétrée par vos soins. Ici vous pourrez choisir votre hébergement, donc décider des capacités de votre blog, créer un design 100 % fait maison ou installer des fonctionnalités supplémentaires qui ne sont pas proposées dans la version par défaut. Elle demande quelques notions techniques mais qui sont assez simples et accessibles pour quiconque s'intéresse un minimum à la "technologie informatique".

Les deux structures sont proches. Il est donc tout à fait possible de commencer avec un blog "autohébergé" chez WordPress.com, puis de migrer vers un blog que vous hébergerez tout seul. Cela vous aidera à comprendre le fonctionnement de WordPress et à vous familiariser avec l'univers des blogs si vous n'en avez jamais eu un auparavant. Enfin, il faut savoir que le fait d'héberger son propre blog représente un coût, alors que l'ouverture d'un blog chez WordPress.com est gratuite. Donc, si vous n'êtes pas encore sûr de ce que vous comptez faire avec votre blog, n'hésitez pas à commencer par un test sur WordPress.com.

Dans cet ouvrage, nous partons du principe que votre choix s'est porté sur la seconde option. Vous y apprendrez donc à installer WordPress, étape par étape, et dès la fin de ce chapitre, votre blog sera en ligne et prêt à accueillir vos articles.

Le kit de départ

Avant de pouvoir utiliser votre blog WordPress, vous allez devoir l'installer. Et pour cela, il vous faut :

- un nom de domaine ;
- un hébergement avec PHP et MySQL ;
- un logiciel FTP ;
- la dernière version de WordPress.

Choisir un nom de domaine

L'accès à un site web ou à un blog se fait par un nom de domaine. Ce domaine possède donc un nom et une extension (.com, .fr, .net, etc.), ce qui donne : *www.nomdomaine.com*. C'est ce qui permettra à n'importe qui possédant un navigateur et une connexion à Internet d'accéder à votre blog. Si le nom de domaine n'est pas indispensable pour installer WordPress, il est tout de même important d'en posséder un dès le départ pour donner un nom et quelque part une "marque" à votre blog.

Choisir un nom de domaine peut paraître simple au premier abord, mais c'est peut-être ce qui va vous demander le plus de temps ici. En effet, vous allez devoir choisir un nom pour le site qui va héberger votre blog. Et ce choix sera différent selon que c'est un blog personnel, vu seulement par votre entourage, un blog avec une thématique particulière, ou encore un blog professionnel. Vous devrez donc prendre en compte différents aspects qui vous permettront de faire le meilleur choix.

Si c'est un blog personnel, vous pouvez utiliser n'importe quel nom, cela n'aura pas beaucoup de conséquences. L'identité du blog se construira autour de ce nom, quel qu'il soit. Vous pouvez utiliser notamment votre nom-prénom.com, le tout attaché, ou encore partir de cette structure et travailler sur des variantes. Vous pouvez également opter pour un nom plus original. Le choix est vraiment large ; mais n'oubliez pas que ce nom de domaine va être votre identité sur le Web. C'est ce nom qui va vous représenter. Il sera très difficile de le modifier par la suite sans qu'il y ait des incidences sur votre référencement et votre positionnement sur Internet. Mais, pour un blog personnel, il n'y a pas vraiment de règles.

Si, par contre, vous souhaitez créer un blog avec une thématique bien précise, il est fortement conseillé de choisir un nom de domaine en relation directe avec cette thématique.

Imaginons, pour illustrer ce propos avec un exemple concret, que vous souhaitiez créer un blog consacré aux voitures anciennes et autres véhicules de collection. Si vous lui attribuez un nom qui n'a rien à voir avec le sujet, vous passez à côté de l'opportunité de donner une identité forte à votre blog. Ce sera notamment le cas sur les moteurs de recherche où les visiteurs potentiels intéressés par votre contenu effectuent des recherches rapides. Un nom visible et bien compréhensible de tous accentuera la probabilité d'avoir des visiteurs sur votre blog guidés par ces moteurs de recherche.

Très souvent, on commence un blog pour se faire plaisir, mais très rapidement on a envie d'être lu et on finit généralement par accorder beaucoup d'attention aux visites en provenance des moteurs de recherche. Et quand on sait qu'ils utilisent le nom de domaine comme mot-clef, on réalise l'importance d'une bonne réflexion avant de se lancer dans la conception d'un blog.

Pour reprendre notre exemple de blog, supposons que vous choisissiez un nom de domaine tel que *www.autos-anciennes.fr*. Un moteur de recherche de type Google prendra en compte les mots-clefs "autos" et "anciennes" pour faciliter les recherches des internautes.

Il peut être tentant de choisir un nom un peu "fun" au début. Mais après quelques mois, si votre blog remporte un certain succès, vous pourriez regretter votre choix, qui sera mal adapté ou pas "très sérieux". Le changer après coup vous fera perdre votre référencement

sur les moteurs de recherche, car ceux-ci auront indexé tous les articles du blog avec votre premier nom de domaine. Vous risquez de perdre en nombre de visites pendant quelque temps, parfois des semaines ou des mois, jusqu'à ce que les moteurs de recherche indexent à nouveau toutes vos pages avec le nouveau nom de domaine.

Une fois que votre choix est arrêté, vous devez acheter ce nom de domaine. La plupart des offres courent sur un an. Au terme de cette année, vous recevrez un e-mail vous demandant si vous souhaitez continuer à posséder ce nom de domaine. Vous allez donc vous procurer votre nom de domaine chez un "registrar", c'est-à-dire une société où vous pourrez enregistrer ce nom de domaine. Nous allons voir par la suite que vous devrez également acheter un hébergement. Et il se trouve qu'aujourd'hui de nombreux hébergeurs vous proposent un pack complet nom de domaine + hébergement. Vous avez donc deux possibilités :

- **Hébergement et nom de domaine chez un même prestataire.** Il vous est alors possible de prendre une solution "tout en un", hébergement + nom de domaine. C'est probablement la solution la plus simple et la moins onéreuse. Cependant, elle peut se révéler risquée chez de petits hébergeurs peu scrupuleux qui ne vous autoriseraient pas par la suite à changer facilement d'hébergement.
- **Hébergement et nom de domaine chez deux prestataires distincts.** Ici, vous avez la possibilité de prendre un hébergement chez un prestataire et un nom de domaine chez un autre. L'avantage, c'est que vous pourrez gérer votre hébergement indépendamment de votre nom de domaine et ainsi en changer facilement. Nous pourrions faire le parallèle avec la téléphonie mobile. Votre numéro de téléphone (ici votre nom de domaine) serait indépendant de votre abonnement (ici votre hébergement), ce qui vous permettrait d'en changer librement sans être lié à votre opérateur téléphonique.

L'inconvénient de cette solution est le coût, puisque la plupart des packs "tout en un" proposent un ou plusieurs noms de domaine inclus gratuitement au moment de l'achat d'un hébergement. Et puis cette seconde solution est plus difficile à mettre en place pour quelqu'un qui n'a pas ou peu de notions d'informatique.

Faites donc le tour des hébergeurs, en voyant notamment les grands noms tels que OVH, 1&1, Amen, ou encore Gandi. Étudiez bien leurs offres et choisissez l'option qui répond le mieux à vos besoins et à votre budget. Les prix pour un nom de domaine seul sont très variables. Tout va dépendre si le nom de domaine est libre ou non. S'il est libre, vous pourrez l'acquérir pour environ 5 euros, à condition que vous le preniez avec une extension .com, .net ou .fr par exemple. Par contre, si vous souhaitez une extension moins répandue, en référence par exemple à votre pays, comme l'extension .ch pour la Suisse, il vous en coûtera un peu plus cher.

Enfin, il existe des noms de domaine qui sont en vente, mais qui ne sont pas libres. Cela veut dire qu'ils ont été achetés pour être revendus. Ce sont souvent des noms de domaine très prisés, ce qui fait qu'ils sont très chers.

Un hébergement

WordPress est un logiciel écrit en PHP dont les données sont stockées dans une base MySQL. De fait, les seules conditions que votre espace d'hébergement doit remplir pour faire fonctionner normalement WordPress sont les suivantes :

- avoir PHP en version 4.3 (ou une version supérieure) ;
- avoir MySQL en version 4.0 (ou une version supérieure).

Notez que, si votre hébergement ne vous propose que des versions plus anciennes des outils cités ci-dessus (par exemple, MySQL v3.23.23), vous pouvez toujours installer WordPress en version 2.0.10. En effet, la branche 2.0.x est stable et doit être maintenue jusqu'en 2010. En revanche, vous ne pourrez pas bénéficier des dernières innovations de WordPress, et certains thèmes et extensions ne marcheront probablement pas comme prévu.

WordPress n'est pas un logiciel qui se lance depuis Windows ou OS X, c'est un script qui s'installe sur un site web. Il est donc indispensable que vous disposiez d'un hébergement web (gratuit ou payant) remplissant les conditions ci-dessus.

Et, une fois de plus, vous allez devoir faire un choix en fonction de vos besoins. Les offres présentées par les prestataires peuvent être très diverses. Il est important de faire le tour des différentes prestations et de bien les comparer.

Alors, au moment de choisir votre hébergement, que faut-il prendre en compte ? Quel va être votre besoin ? De par notre expérience de blogueurs, nous dirions qu'il y a quatre aspects à considérer :

- **Le type d'hébergement.** Quand vous visitez le site web d'un hébergeur, vous vous rendez rapidement compte qu'il existe une multitude d'offres et surtout différents types d'hébergements. Vous comprendrez aisément en les examinant un par un qu'ils dépendent complètement du type de site que vous allez créer. De manière générale, vous trouverez trois grandes familles d'hébergements :
  - **Hébergements mutualisés.** Ils sont les plus utilisés car ils sont peu chers et faciles à installer et à gérer. Dans cette option, votre hébergement va partager un serveur avec d'autres sites web ou blogs (idéal pour les petits budgets et blogs personnels).
  - **Hébergements dédiés.** Ici, vous avez un serveur, donc une machine physique, entièrement "dédié" à votre site. Ce type d'hébergement est principalement destiné à ceux qui veulent maîtriser entièrement la gestion du serveur qui héberge leur blog. Il apporte également plus de puissance. Cependant, ce sont les offres les plus chères (conçues pour les gros sites et pour ceux qui administrent eux-mêmes le serveur).
  - **Hébergements virtualisés ou privés.** La virtualisation est à mi-chemin entre les deux solutions présentées ci-dessus. Dans ce cas précis, vous devrez partager un serveur avec d'autres sites, mais vous serez complètement isolé et votre environnement se comportera exactement comme celui d'un serveur dédié. Vous devrez également administrer votre site vous-même. Les prix se situent entre ceux d'un hébergement mutualisé et ceux d'un serveur dédié (solution intéressante pour ceux qui trouvent l'offre mutualisée trop restreinte mais qui ne veulent pas investir dans un serveur dédié).

À vous donc de faire votre choix en fonction de vos besoins et de votre portefeuille.

- **La capacité de stockage.** Quel que soit le type d'hébergement, vous pouvez avoir des capacités de stockage très différentes. Cela dit, aujourd'hui, les capacités proposées suffisent largement pour héberger un blog. Normalement, vous ne devriez avoir aucun souci avec l'offre que vous choisirez. Un blog WordPress ne prend que 5 Mo. Ce sont surtout tous les fichiers type photos, images, voire podcasts et vidéos, qui prendront le plus de place. Ici encore, tout dépendra du contenu de votre blog, et bien souvent vous aurez tendance à héberger certains types de médias comme les vidéos sur des sites tels que YouTube ou DailyMotion.
- **La bande passante.** C'est également une donnée à prendre en compte dans le choix de son hébergement, bien que les offres semblent s'orienter vers une bande passante illimitée. Cependant, ce n'est pas encore le cas de tous les prestataires, notamment pour les offres d'entrée de gamme. Il est donc important de bien savoir ce que c'est et quelle quantité vous sera probablement nécessaire.

  La bande passante correspond à la largeur du "tuyau" qui relie le site au reste d'Internet et aussi à la quantité de données qui vont transiter chaque mois. Plus les blogs seront riches en images et en vidéos, plus le besoin en bande passante sera important. En général, pour un site "normal", c'est-à-dire sans trop de photos ou vidéos, on peut tabler sur une capacité mensuelle de 15 Go, et cela peut aller jusqu'à 60-80 Go pour les blogs les plus gros.
- **La taille de la base de données.** Beaucoup de prestataires proposent des bases de données qui ne sont pas très importantes. La base de données, c'est l'endroit où sera placé le contenu de votre blog. C'est là que vont être stockés vos articles, vos commentaires, vos pages, etc. Cette base de données est souvent utilisée par des extensions de WordPress, comme les outils de statistiques. Il est donc important ici d'avoir, dès le départ, une base de données suffisamment grande pour pouvoir travailler convenablement. Pour notre part, nous vous conseillons de ne pas prendre d'hébergement avec une base de données dont la taille serait inférieure à 25 Mo. Vous risqueriez d'être rapidement bloqué, surtout si vous installez des extensions qui vont utiliser votre base de données. Le mieux serait même d'avoir plusieurs bases de données pour séparer les tables de votre blog de celles d'éventuelles extensions ou services extérieurs.

Prenez donc le temps de faire le tour des offres et de choisir celle qui vous convient le mieux. De toute façon, vous aurez toujours la possibilité par la suite d'évoluer vers une autre offre, soit chez le même prestataire, soit chez un nouveau. Pour débuter, il existe des solutions à 1 euro par mois qui sont grandement suffisantes pour un blog qui publie un article chaque jour, avec peu de vidéos.

Afin de vous aider dans votre choix, voici une liste non exhaustive des prestataires que nous vous conseillons :

- OVH (**www.ovh.net**) ;
- 1&1 (**www.1&1.fr**) ;

- Gandi (**www.gandi.net**) ;
- Infomaniak (**www.infomaniak.ch**).

Évitez en revanche les prestataires suivants :

- Online ;
- Free.

Nous vous les déconseillons car ils proposent des solutions gratuites qui ne vous permettront pas de profiter au maximum de votre blog. Ils imposent en effet des règles techniques qui se révèlent parfois contraignantes pour un blog.

Si vous n'arrivez pas à vous décider, n'hésitez pas à vous tourner vers la communauté francophone de WordPress (**www.wordpress-fr.net**), qui saura vous aider et vous orienter selon vos besoins.

Vos données de connexion

Une fois que vous avez choisi votre hébergement, vous allez recevoir différents e-mails provenant de votre hébergeur. Il s'agit des données de connexion à votre hébergement, ainsi que celles concernant la connexion à la base de données.

Voyons quelles sont les informations qui doivent vous être transmises pour pouvoir procéder à l'installation de votre blog WordPress :

- Informations de connexion à l'hébergement :
  - nom du serveur FTP ;
  - nom de l'utilisateur ;
  - mot de passe de l'utilisateur.
- Informations de connexion à la base de données :
  - nom du serveur de la base ;
  - nom de la base ;
  - nom de l'utilisateur de la base ;
  - mot de passe de l'utilisateur.

Un logiciel FTP

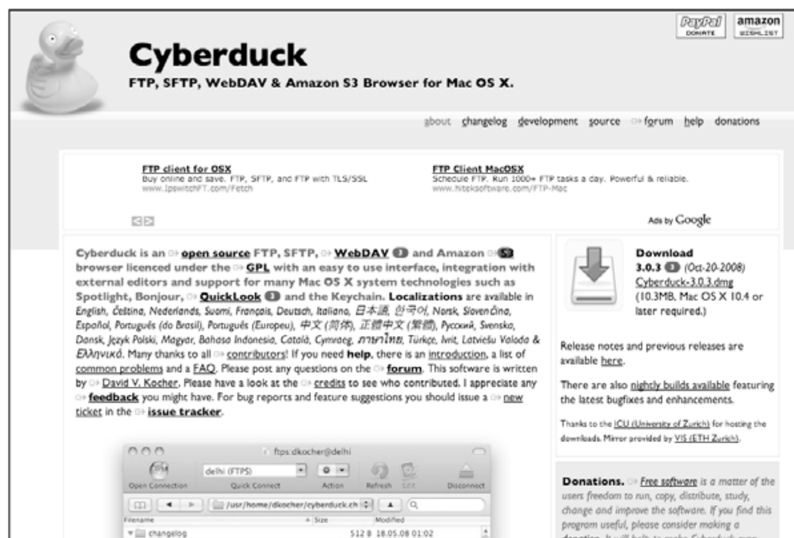
Pour vous connecter à votre hébergement, vous aurez besoin d'un logiciel FTP qui utilisera les informations récoltées précédemment auprès de l'hébergeur.

De nombreux logiciels FTP sont disponibles sur Internet. Encore une fois, il n'est pas toujours facile de faire le bon choix. Sachez tout de même qu'il en existe certains qui font très bien leur travail et qui sont gratuits. C'est le cas notamment de FileZilla (**http://filezilla-project.org/**), qui existe aussi bien pour Windows, Mac OS X et Linux, ou encore le très connu Cyberduck (**www.cyberduck.ch**), qui fonctionne uniquement sur Mac OS X.

Figure 2.01
FileZilla.



Figure 2.02
Cyberduck.



Dernière version de WordPress

Maintenant que vous avez tous les outils pour créer votre blog, il ne vous reste plus qu'à récupérer la dernière version de WordPress. Pour cela, rendez-vous sur www.wordpress-fr.net et cliquez sur le bouton Télécharger WordPress. Le téléchargement démarre automatiquement.

Une fois que le téléchargement est terminé, décompressez l'archive (dossier récupéré au format .zip). Vous obtenez alors un dossier intitulé "wordpress". En l'ouvrant, vous verrez qu'il contient l'ensemble des fichiers qui composent votre futur blog WordPress. Ce sont ces fichiers que vous devrez transférer sur votre hébergement.

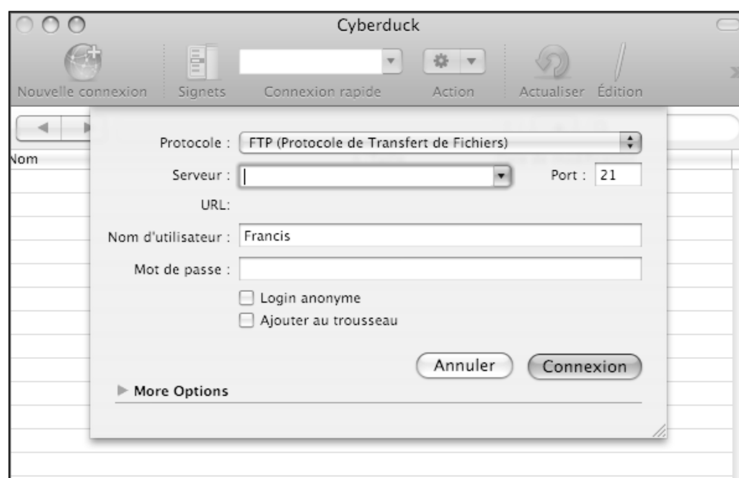
Installation de WordPress

Votre kit de départ est prêt, vous allez pouvoir procéder à l'installation même de votre futur blog WordPress. Cette procédure va se dérouler en plusieurs étapes et ne va durer que cinq petites minutes !

Transfert des fichiers WordPress sur votre hébergement

1. Ouvrez votre logiciel FTP et choisissez une nouvelle connexion (voir Figure 2.03).

Figure 2.03
Connexion FTP.



2. Renseignez les informations suivantes, fournies par votre hébergeur (voir Figure 2.04) :
   – nom du serveur ;
   – nom de l'utilisateur ;
   – mot de passe de l'utilisateur.

   (D'autres informations seront aussi à compléter. Normalement, elles devraient être déjà remplies par défaut. Il s'agit du port, ici 21 par défaut, et du protocole à utiliser. Ici, c'est le protocole FTP.)
3. Une fois que ces informations sont saisies, connectez-vous. La fenêtre d'accès sera différente selon le prestataire que vous aurez choisi. Chez certains, vous devrez ouvrir un dossier "www" pour stocker l'ensemble de vos données (voir Figure 2.04), alors que chez d'autres, ce sera un dossier "web", "public_html", ou encore vous arriverez directement dans l'espace où vous allez transférer vos fichiers (historiquement, ce dossier s'appelait "httpdoc"). Avant de procéder à un transfert complet de vos fichiers et dossiers, n'hésitez pas à regarder une nouvelle fois les e-mails envoyés par votre hébergeur. Généralement, cette information y est précisée. Si ce n'est pas le cas, contactez-le rapidement.

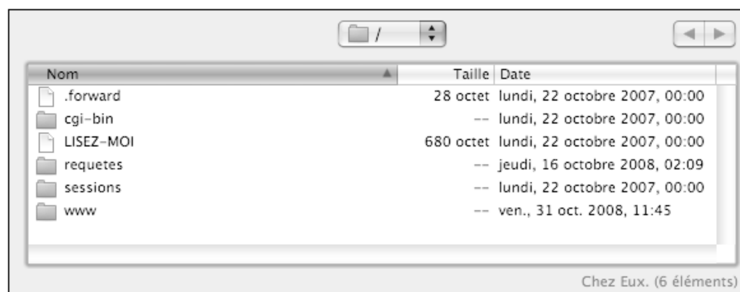


Figure 2.04
Fenêtre de connexion OVH.

4. Transférez dans ce dossier "web" l'ensemble du dossier WordPress que vous avez téléchargé précédemment (voir Figure 2.05).

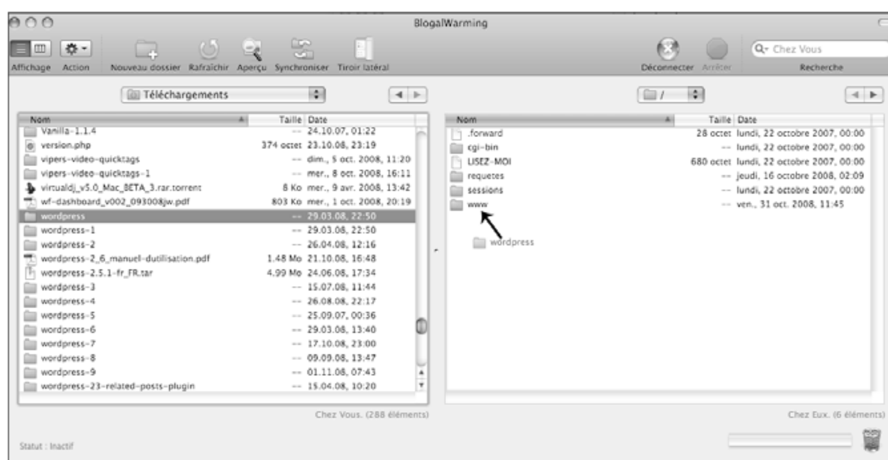


Figure 2.05
Transfert des fichiers.

Tous vos fichiers sont désormais en ligne. Il ne vous reste plus qu'à installer WordPress. Cette installation va se faire par le biais de votre navigateur Internet (tout ordinateur en possède forcément un !).

L'installation va se passer en deux temps :

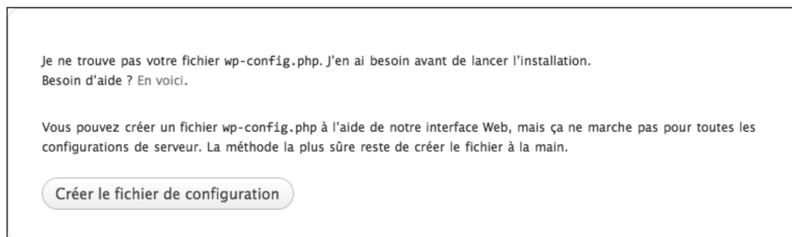
1. **Création de la base de données de votre blog.** WordPress va commencer par se connecter à la base de données de votre hébergement et va créer les tables qui seront utilisées pour votre blog.

2. **Création du blog.** Une fois les tables générées, vous pourrez créer votre blog en donnant les premières informations de son identité.

Création de la base de données WordPress

Ouvrez votre navigateur Internet et saisissez l'URL de votre blog. Une fenêtre apparaît et vous présente les informations nécessaires pour commencer l'installation de WordPress.

Figure 2.06
Écran 1 installation WordPress.



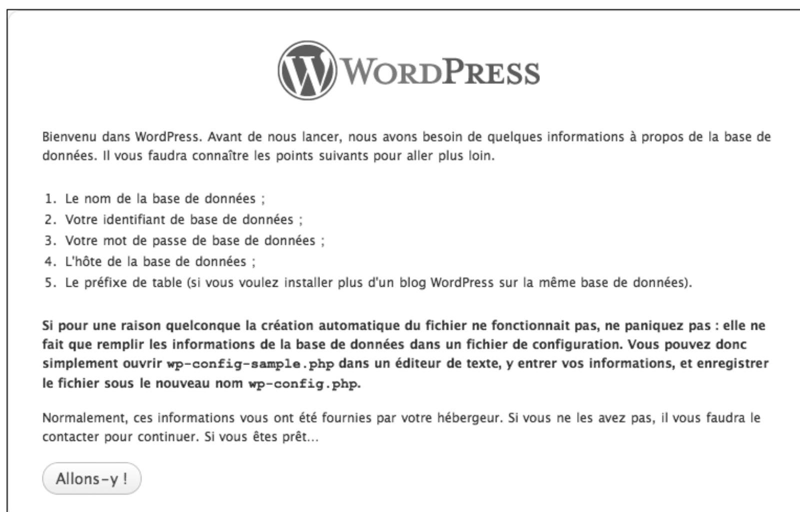
Sur cet écran (voir Figure 2.06), un message vous informe que pour accéder à la base de données il vous faut créer un fichier nommé wp-config.php. Ce fichier va contenir les informations de connexion à la base de données. Et c'est grâce à ces informations que WordPress va pouvoir aller sur votre base et générer les tables nécessaires à la création de votre blog.

Wordpress vous informe ici que vous avez deux manières de créer ce fichier : automatiquement, *via* l'interface web, ou manuellement :

- **Création automatique du fichier config.php.** Ici vous allez renseigner les informations de connexion à la base de données *via* votre navigateur web.

  1. Lorsque vous êtes sur la première page d'installation de WordPress, cliquez sur le bouton qui vous permet de créer le fichier config.php. Vous arrivez sur une nouvelle page qui vous présente les informations dont vous allez avoir besoin pour créer ce fichier wp-config.php (voir Figure 2.07). Une fois que vous avez lu et compris ces informations, cliquez sur Allons-y !

Figure 2.07
Écran 2 installation WP.



2. Une nouvelle fenêtre s'ouvre et vous propose de renseigner les informations de connexion à votre base de données – informations identiques à celles présentées dans la méthode manuelle (voir Figure 2.08).

Figure 2.08
Écran 3 Installation WP.

WordPress

Entrez ci-dessous les détails de connexion à votre base de données. Si vous ne les connaissez pas avec certitude, contactez votre hébergeur.

Nom de la base de données	wordpress	Le nom de la base dans laquelle vous voulez installer WP.
Identifiant	username	Votre identifiant MySQL.
Mot de passe	password	...et votre mot de passe MySQL.
Hôte de la base de données	localhost	Si localhost ne marche pas, vous devrez demander cette information à votre hébergeur.
Préfixe de table	wp_	Si vous voulez installer plusieurs blogs WordPress dans une même base de données, modifiez ce champ.

Valider

3. Sur cette page, vous pouvez choisir un préfixe pour les tables de la base de données. Par défaut, le préfixe "wp_" est proposé. Nous allons prendre ici un exemple concret pour bien comprendre ce que sont les préfixes de la base de données.

   WordPress fonctionne avec une base de données. Chacune des informations du blog, comme les articles, les commentaires, les utilisateurs et bien d'autres données, va être stockée dans des tables. Il va donc y avoir une table pour les articles, une pour les commentaires et ainsi de suite. Ces tables ont un nom. Mais ce nom est unique. Alors, imaginons que vous ayez envie d'utiliser la même base de données pour plusieurs blogs ; il va vous falloir différencier ces tables, puisqu'au départ elles portent le même nom. C'est là que le préfixe va être important car il va être ajouté devant le nom de la table. Ainsi, si vous prenez wp_ comme préfixe, votre table d'articles va s'intituler wp_posts. Dès lors que vous créerez un autre blog et utiliserez la même base de données, vous devrez penser à modifier le préfixe, en "wp2_" par exemple. Ainsi, les tables de vos différents blogs seront différenciées et il n'y aura aucun risque de conflit.

   Cependant, dans l'absolu, il vaut mieux installer plusieurs blogs WordPress sur différentes bases de données pour éviter les conflits.

   Dans le cas présent, nous partons du principe que nous installons notre premier blog, nous gardons donc le préfixe proposé par défaut.

4. Une fois que toutes les informations sont renseignées, cliquez sur le bouton de validation. Une nouvelle fenêtre apparaît pour confirmer que les tables ont bien été créées et que nous allons maintenant pouvoir installer l'ensemble du blog WordPress (voir Figure 2.09).

Figure 2.09
Écran 4 installation WP.



Cliquez une nouvelle fois en bas de la page pour procéder à l'installation du blog.

- **Création manuelle du fichier config.php.**

1. Renommez le fichier wp_config_sample.php en wp_config.php.
2. Ouvrez le fichier wp_config.php et trouvez les lignes suivantes :

```
define('DB_NAME', 'putyourdbnamehere'); // Le nom de la base de données
define('DB_USER', 'usernamehere');     // Votre identifiant MySQL
define('DB_PASSWORD', 'yourpasswordhere'); // ...et votre mot de passe
define('DB_HOST', 'localhost');     // Dans la plupart des cas, vous n'aurez
pas à modifier cette ligne
```

3. Remplacez les informations de `putyourdbnamehere` (mettez votre nom de base de données ici), `usernamehere` (nom d'utilisateur ici), `yourpasswordhere` (mot de passe ici) et `localhost` (nom du serveur) par vos données personnelles de connexion à la base de données :
   – nom de la base (`DB_NAME`) ;
   – nom de l'utilisateur de la base (`DB_USER`) ;
   – mot de passe de l'utilisateur (`DB_PASSWORD`) ;
   – nom du serveur de la base (`DB_HOST`).
4. Un peu plus bas dans le fichier, vous retrouvez le préfixe à attribuer à la base de données, dont nous avons parlé un peu avant :

```
// You can have multiple installations in one database if you give each
➥a unique prefix
$table_prefix  = 'wp_';   // Only numbers, letters, and underscores please!
```

Vous allez également y trouver les clefs de sécurité `AUTH_KEY`, `SECURE_AUTH_KEY`, `LOGGED_IN_KEY`, `NONCE_KEY`, `AUTH_SALT`, `SECURE_AUTH_SALT`, `LOGGED_IN_SALT`, `NONCE_SAL`, qui permettront de mieux protéger les informations disponibles dans les cookies des utilisateurs.

Ces clefs sont définies par défaut mais vous pouvez très bien les modifier à votre convenance. Voici un exemple donné :

```
define('AUTH_KEY',         'PnIOV~sv:Iix,^p7UOJt1:1NnqG@.M<|bEAq*sCgG*4S$9?|
➥X@bFBhB*6&?Y=FSp');
define('SECURE_AUTH_KEY',  'ZW!aug}N*&7[..q*pEMDH2:wURz%J49-&Ct:`
➥=qyK/=Jt(pPA[XyoQhU($hnTX8R');
define('LOGGED_IN_KEY',    '0woykcp`9BAafuW^1j15s9W-x_sNHhA<SRyZN?Nia/PB
➥[j8L2ne,K52Mm67-qsZ2');
define('NONCE_KEY',        '-8.Yuj3ZWJ?~[{RjEx%4*`m-A<UzRwSi/YNq,
➥*N nSwau$So;.`5PQQNb]I.Z=Qe');
define('AUTH_SALT',        'Bz;Uu(VFzU^@8OoF-*.Pzp=9i;t~%Q8nY%dy#>@5<+s$]
➥~m+9Ur^^HJ z4L09B$$');
define('SECURE_AUTH_SALT', 'T8(-OJ*]vTgX{%3K*ghnsP&l/OUJ53eCx ty:^bE#e!z-r.
➥dxu_8k}Cxkj1Zi#R!');
define('LOGGED_IN_SALT',   'SwU*wa=U!T%*hgHvy5s`jQ7T7Do]dY8?8Zta`5MLr8C3.
➥SL!=04hnxs%1U{`S.,O');
define('NONCE_SALT',       'yV@^k:s`3;LU9*B_.}SDFf&s)ytEr2PdS`uZ/jCt<9Y<
➥Lwt1|;4#:q*kFM@&I^s<');
```

5. Une fois les informations saisies, enregistrez le fichier et allez à l'adresse **www.exemple.fr/wp-admin/install.php** pour valider la création des tables. C'est tout !

Création du blog

Que vous ayez créé le fichier de configuration wp_config.php manuellement ou automatiquement, vous arrivez dans tous les cas sur la fenêtre de création de votre blog (voir Figure 2.10). Ici, vous devrez renseigner quelques informations de base qui vont être utilisées pour créer votre blog :

Figure 2.10
Écran 5 installation WP.

- **Titre du blog.** Saisissez le titre que vous avez choisi pour votre blog. Il n'y a rien de définitif, cependant, vous pourrez toujours le modifier par la suite.
- **Nom utilisateur.** C'est le nom que vous allez utiliser pour vous représenter sur votre site.
- **Mot de passe.** Il accompagne votre nom d'utilisateur et vous sera demandé à chaque fois que vous souhaiterez vous connecter à l'administration de votre blog.
- **Adresse e-mail.** Saisissez l'adresse e-mail qui sera utilisée pour l'administration du blog. C'est aussi une donnée que vous pourrez modifier plus tard, mais vous allez en avoir rapidement besoin puisque WordPress va vous envoyer un e-mail avec votre mot de passe à l'interface d'administration, et ce dès que vous aurez créé votre blog.
- **Indexation du blog dans les moteurs de recherche.** À vous de décider si vous voulez que les pages de votre blog apparaissent dans ces outils. Si vous souhaitez rapidement trouver un public, vous avez tout intérêt à cocher cette case. Par contre, si vous souhaitez avoir votre blog en ligne mais le garder plutôt "privé", ne cochez pas cette case. Notez cependant que quand je dis privé, cela veut seulement dire que, même si le blog reste visible par tout le monde, vous n'aurez pas de visites en provenance des moteurs de recherche.

  Une fois encore, sachez qu'il vous sera possible de modifier votre choix par la suite, rien n'est définitif.

La destinée de votre blog étant tracée, cliquez sur le bouton Installer WordPress.

Figure 2.11
Écran 6 installation WP.

Vous pouvez maintenant vous connecter à votre blog, tout fraîchement installé ! Accédez à la page d'identification en cliquant sur le lien qui vous est proposé et entrez votre identifiant ainsi que le mot de passe (voir Figure 2.12).

Figure 2.12

Connexion.

WordPress

Identifiant

Mot de passe

Se souvenir de moi

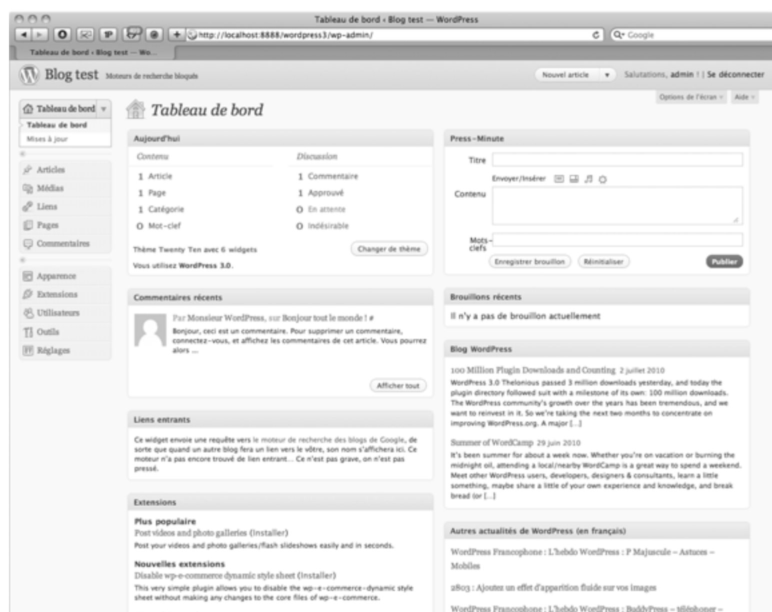
Se connecter

Mot de passe oublié ?

Validez le tout. Vous êtes maintenant à bord (voir Figure 2.13) ! Bienvenue sur WordPress !

Figure 2.13

Administration WordPress.



Histoire de bien voir que votre blog existe réellement, cliquez sur le titre du site, situé en haut à gauche. Une nouvelle fenêtre s'ouvre avec votre nom de domaine et votre blog (voir Figure 2.14). Tout est prêt ! Votre blog n'attend plus que vous lui donniez vie !

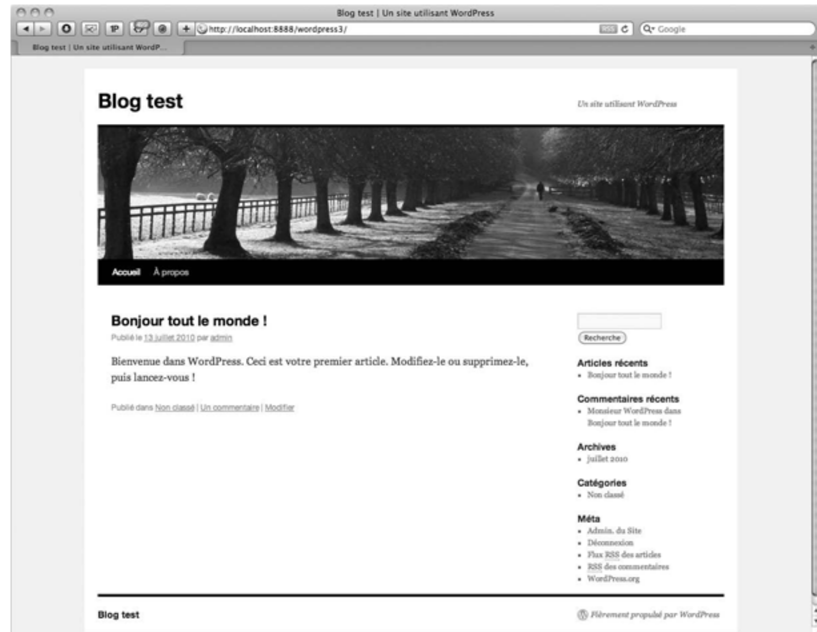


Figure 2.14
Écran blog WP.

3 Le quotidien du blogueur

LE CAMPUS

Premiers pas avec WordPress

Dans ce chapitre, vous allez découvrir ce qui va rapidement devenir votre quotidien de blogueur. Vous allez apprendre à rédiger des articles, créer des catégories, insérer des images, des vidéos, etc. Vous apprendrez également à gérer votre blog et à faire face aux attaques extérieures comme le spam dans les commentaires. Cette partie est le cœur de WordPress.

Tout le cheminement sera articulé autour d'un article. Vous apprendrez à le rédiger mais aussi à prendre en compte toutes les options qui l'entourent.

Découverte de l'interface

La connexion

Tout d'abord, pour pouvoir utiliser votre blog, vous allez vous connecter à son interface d'administration. Pour cela, connectez-vous à l'URL suivante : **http://www.exemple.fr/wp-login.php** (voir Figure 3.01).



Figure 3.01
Page de connexion.

Vous arrivez sur le tableau de bord de WordPress, qui est en quelque sorte la "page d'accueil" de l'interface d'administration de votre blog (voir Figure 3.02).

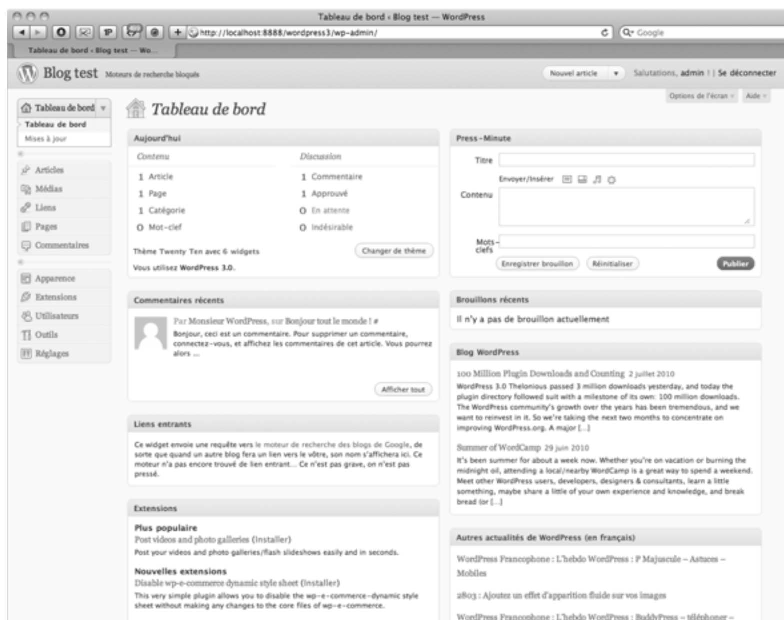


Figure 3.02

Tableau de bord WordPress.

Si vous êtes déjà connecté, vous pouvez accéder directement à cet écran en utilisant l'URL suivante : **http://www.exemple.fr/wp-admin**.

Anatomie du tableau de bord

Comme nous l'avons vu précédemment, lorsque vous vous connectez à l'interface d'administration de WordPress, vous arrivez directement sur le tableau de bord. Cette page regroupe différentes informations qui vous permettront d'avoir une vision globale de ce qui se passe sur votre blog, et elle vous donne également la possibilité d'accéder directement à des fonctions importantes pour notamment créer une nouvelle page, un nouvel article ou encore gérer vos commentaires (voir Figure 3.03).

Cette page est facilement paramétrable pour faire apparaître les informations qui vous semblent les plus utiles. Pour cela, vous avez un petit bouton, situé en haut à droite de l'écran, intitulé Options de l'écran, qui vous permet de décider quels blocs ou modules faire apparaître sur le tableau de bord.

Ces "options de l'écran" vont se retrouver sur une grande partie des pages de l'interface d'administration de WordPress. Elles vous permettront aussi de cacher les informations qui ne vous seront pas utiles (voir Figure 3.04).

Lorsque vous avez choisi les différents modules qui vont apparaître, vous pouvez décider de leur ordre tout simplement en plaçant la souris dessus et en les déplaçant avec le bouton de la souris enfoncé (voir Figure 3.05).

Figure 3.03

Tableau de bord.

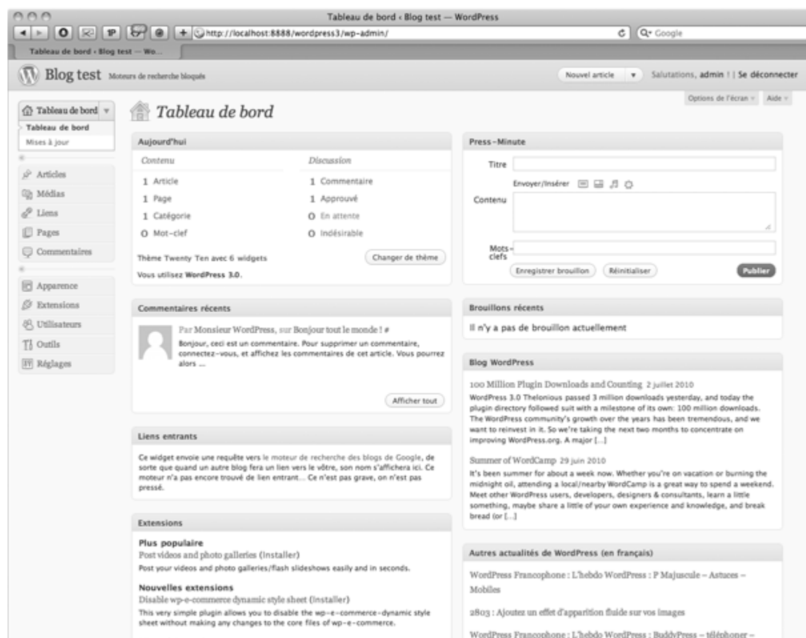


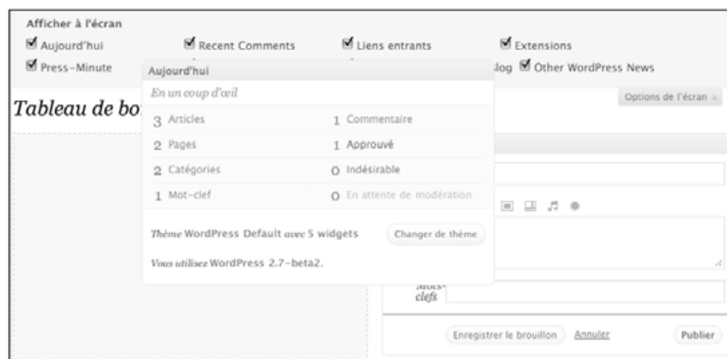
Figure 3.04

Options du tableau de bord.



Figure 3.05

Déplacement des modules du tableau de bord.



À droite des options de l'écran se trouve un deuxième bouton, Aide, qui permet d'accéder directement à l'aide en ligne de WordPress, c'est-à-dire au forum de discussion ainsi qu'au Codex.

Sur la gauche du tableau de bord se trouve le menu de navigation de l'interface d'administration de WordPress (voir Figure 3.06). Vous y trouvez trois blocs distincts :

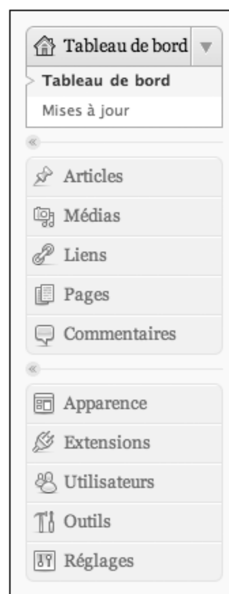


Figure 3.06
Menu de navigation de l'interface de WordPress.

- **Accès au tableau de bord.** Pour y revenir à partir de n'importe quelle page.
- **Articles.** Ce bloc regroupe toutes les fonctions de rédaction et de gestion des articles mais aussi des pages du blog. Vous y retrouvez également la gestion des médias et des commentaires.
- **Gestion et paramétrage du blog.** Ici, vous définissez l'apparence de votre site en choisissant un thème, vous y ajoutez des extensions, vous pouvez créer de nouveaux utilisateurs et réglez les différents paramétrages pour le bon fonctionnement de votre blog.

En haut du tableau de bord, vous trouvez également un autre menu, dans une bande grisée sombre, qui vous permet de vous déconnecter de l'interface d'administration, mais qui sert également de raccourci pour accéder aux pages de rédaction d'articles et aux pages de gestion des commentaires (voir Figure 3.07).



Figure 3.07
Menu de navigation horizontal.

Votre premier article

Introduction

L'intérêt principal d'un blog se concentre surtout autour de la possibilité d'écrire des billets, des articles. Ici, nous allons donc voir dans le détail comment rédiger ces articles, comment bien les paramétrer mais aussi comment les gérer une fois qu'ils sont publiés et en ligne.

Pour rédiger votre premier article, cliquez sur l'onglet Articles dans le menu de navigation situé à gauche de la page web. Une nouvelle fenêtre s'ouvre sur la liste des articles. Pour le moment, il n'y a que l'article par défaut de WordPress. À gauche, sous l'onglet Articles, vous allez trouver des sous-menus. Cliquez sur Ajouter un nouvel article. Vous êtes maintenant sur la page de rédaction d'un article (voir Figure 3.08). Sur cette page, vous avez toute une série d'informations que nous allons détailler au fur et à mesure de la rédaction de votre article.

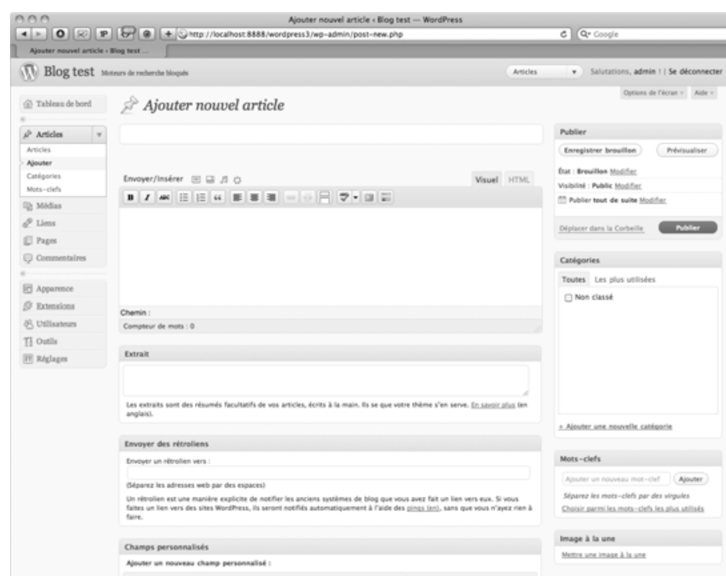


Figure 3.08
Page de rédaction d'un article.

Tout commence par le titre !

Le titre d'un article a beaucoup d'importance parce qu'il conditionne la lecture de son contenu (voir Figure 3.09). C'est le titre qui va retenir ou non l'attention du lecteur. Il faut donc lui apporter une attention toute particulière. Sans être nécessairement accrocheur, il doit déjà donner une idée du sujet dont vous allez parler par la suite.

Vous allez le saisir directement dans le cadre réservé à cet effet, sous l'indication Ajouter un nouvel article.

Figure 3.09
Le titre de l'article.



Cependant, il n'est pas obligatoire de trouver le bon titre avant de rédiger l'article. Vous pouvez en choisir un temporairement qui vous aidera à identifier l'article mais que vous pourrez changer par la suite. Finalement, c'est souvent le contenu même qui vous aidera à trouver le meilleur titre.

Le titre est très important parce que c'est lui qui va conditionner l'intérêt du visiteur. Par exemple, si celui-ci passe sur votre blog pour un autre article et décide de prolonger sa visite, nul doute que s'il voit un titre d'article intéressant, il va essayer d'en savoir plus en lisant le contenu. C'est la même chose pour les moteurs de recherche. Il est probable que vous ayez au minimum quelques visites en provenance de ces moteurs, et quand un internaute cherche quelque chose, il veut rapidement le trouver. Plus votre titre sera explicite et donnera une bonne description du contenu et plus il aura d'impact.

Le contenu de votre article

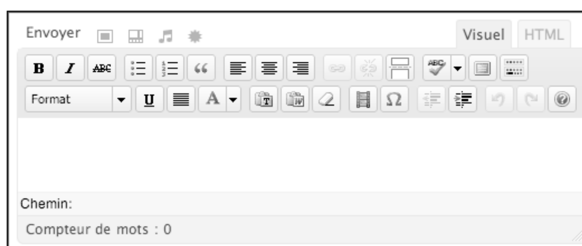
Le contenu, c'est le cœur de l'article. C'est par lui que vous allez faire passer vos messages. À vous de trouver les mots pour vous exprimer, les bonnes phrases qui ont du sens pour essayer de partager vos pensées. Le plus souvent, un style direct, clair, et des articles ne dépassant pas les 1 000 mots sont les plus appréciés. Évitez également les phrases et les paragraphes trop longs. Le style doit être fluide et lisible rapidement par l'internaute.

Pour vous aider à rédiger vos articles dans les meilleures conditions, WordPress vous propose deux éditeurs de texte avec des fonctionnalités différentes, selon les habitudes et les besoins de chacun. Il s'agit de l'éditeur visuel et de l'éditeur HTML.

L'éditeur visuel

C'est ce qu'on appelle un éditeur WYSIWYG (*What You See Is What You Get*). Ce que vous allez voir dans l'éditeur est ce qui s'affichera globalement sur votre blog (voir Figure 3.10). Vous retrouvez ici le concept et l'ergonomie classique d'un logiciel de traitement de texte, mais modifiés pour coller au plus près avec les besoins de la rédaction d'un article sur un blog. Hormis les boutons classiques de mise en page, vous avez donc quelques options qui vous permettront de personnaliser au mieux vos articles et en même temps votre blog.

Figure 3.10
Éditeur visuel.



Le Tableau 3.01 fait un récapitulatif des différentes fonctions de l'éditeur visuel.

Tableau 3.01 : Contenu de l'éditeur visuel

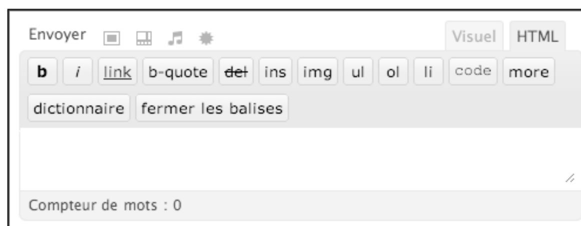
Boutons	Fonctions
B	Met le texte en caractères gras.
I	Met le texte en italique.
ABC	Permet de barrer le texte.
	Crée une liste à puces non ordonnée.
	Crée une liste ordonnée.
“	Permet de mettre en relief une citation (le thème doit avoir pris en compte la balise `blockquote`).
	Aligne le texte à gauche.
	Aligne le texte à droite.
	Centre le texte.
	Permet de créer un hyperlien dans votre article.
	Permet de supprimer l'hyperlien préexistant.
	Permet de ne faire apparaître que la première partie de l'article (avant d'appuyer sur le bouton) sur la première page du blog, avec un lien Lire la suite. Le paramétrage de cette fonction est expliqué en détail dans la partie de ce livre consacrée à la création d'un thème pour WordPress.
ABC	Correcteur d'orthographe. En cliquant sur la flèche, vous pouvez choisir la langue que vous voulez.
	Permet de rédiger un article en mode Plein écran.
	En cliquant sur ce bouton, vous accédez à une deuxième ligne de fonctionnalités de l'éditeur visuel.

Boutons	Fonctions
Paragraphe	Définit le style pour chaque élément de votre article, que ce soit les paragraphes, les titres de parties ou encore le code.
	Permet de souligner le texte.
	Aligne le texte à gauche et à droite.
	Permet de définir une couleur pour votre texte.
	Colle du texte simple, issu de n'importe quel éditeur de texte.
	Colle du texte en provenance de Microsoft Word.
	Efface la mise en forme de tout ou partie du texte.
	Insère une vidéo dans votre article. Cette fonction sera détaillée dans la section "Insérer un média".
	Permet d'insérer des caractères particuliers dans un article.
	Permet de désindenter un texte qui aurait été indenté.
	Permet d'indenter un texte vers la droite.
	Annule une modification.
	Rétablit une modification.
	Aide de l’éditeur visuel.

L'éditeur HTML

Si l’éditeur visuel pour WordPress est très complet, il en est autrement pour l’éditeur HTML (voir Figure 3.11). Cependant, celui-ci ne s’adresse pas au même public. Ici, si vous n’avez aucune notion en HTML, passez votre chemin. Par contre, si vous avez la fibre "développeur" sans pour autant l’être et si vous voulez contrôler tous les aspects de vos articles, vous utiliserez rapidement l’éditeur HTML.

Figure 3.11
Éditeur HTML.



Ici, vous allez avoir le texte tel qu'il est compris et assimilé par le navigateur web, alors que *via* l'éditeur visuel, le texte est tel qu'il va apparaître dans le navigateur web. Le fond est le même, c'est la forme qui est différente.

Prenons un exemple simple pour illustrer nos propos. Tapez un mot dans l'éditeur HTML. Vous allez le surligner avec votre souris et cliquer sur le bouton `b` qui est le symbole HTML pour dire "gras". Vous pouvez voir que maintenant le texte est entouré par une balise `strong` qui est la balise `html` pour mettre des portions de texte en gras (voir Figure 3.12).

Figure 3.12
Texte dans l'éditeur HTML.



Si maintenant vous passez du côté de l'éditeur visuel en cliquant sur le lien Visuel en haut à droite, vous découvrez le même texte mais ici directement en gras, tel que vous le verrez sur le blog. Vous pouvez également faire le test en sens inverse ; vous voyez que le résultat est différent alors que le texte est le même (voir Figure 3.13).

Figure 3.13
Texte dans l'éditeur visuel.



Pour l'éditeur HTML, les options sont moins nombreuses mais largement suffisantes pour rédiger facilement et rapidement un article (voir Tableau 3.02).

Tableau 3.02 : Contenu de l'éditeur HTML

Balises	Fonctions
`b`	Met le texte en caractère gras.
`i`	Met le texte en italique.
`b-quote`	Permet de mettre en relief une citation (le thème doit avoir pris en compte la balise `blockquote`).
`del` (barré)	Permet de barrer le texte.
`ins`	Permet de mettre en relief du texte qui aurait été ajouté après publication de l'article.
`img`	Permet d'insérer simplement une image dans votre article. Il vous suffit de saisir l'URL de l'image et de lui donner une description.
`ul`	Crée une liste à puces non ordonnée.
`ol`	Crée une liste ordonnée.
`li`	Permet d'ajouter une ligne à une liste.
`code`	Permet d'identifier des exemples de code cités dans votre article et de les séparer du code de la page web.
`more`	Fonction "more" dont le principe a été présenté plus haut.
`lookup`	Permet d'obtenir une définition pour un mot précis.
`close tags`	Permet de fermer les balises que vous auriez laissées ouvertes.

Bien entendu, les balises proposées ici sont celles qui sont le plus souvent utilisées, mais vous pouvez très bien vous servir de toutes les autres balises HTML qui répondent plus spécifiquement à votre besoin.

Insérer un média

Avec les deux éditeurs présentés ci-dessus, vous avez toutes les cartes en main pour proposer des articles complets du point de vue du contenu mais aussi au niveau de la mise en forme. Maintenant, nous allons encore plus loin en vous montrant comment insérer un média dans vos articles.

Ces médias peuvent être de différentes sortes. Il peut s'agir de photos, de vidéos ou encore de fichiers audio. Les deux éditeurs vous proposent par défaut d'ajouter des médias. L'éditeur visuel propose d'insérer des vidéos, alors que l'éditeur HTML vous permet d'insérer des images.

Mais, depuis la version 2.5, WordPress possède une bibliothèque de médias qui vous permet de transférer tout type de fichier média depuis votre ordinateur sur le serveur de votre blog, mais aussi de les gérer directement à partir de l'interface d'administration de WordPress. Et c'est le gros avantage par rapport aux méthodes utilisées par les deux éditeurs.

Voyons dans le détail toutes les possibilités offertes pour insérer un média dans vos articles.

La bibliothèque de médias

Si vous avez bien observé la page de rédaction d'un article, vous avez sûrement remarqué un lien intitulé Envoyer, situé juste au-dessus de l'éditeur, et suivi de quatre petits pictogrammes représentant différents types de médias (voir Figure 3.14).



Figure 3.14
Insertion d'un média.

Chaque bouton va ouvrir une nouvelle fenêtre qui vous permettra d'insérer un média.

Insertion d'une image

Il faut que vous ayez au préalable une image à ajouter dans votre article. Celle-ci peut se trouver sur votre ordinateur ou sur un serveur en ligne.

Pour insérer une image dans votre article, vous allez tout d'abord pointer le curseur de votre souris à l'endroit où vous souhaitez l'intégrer.

Ensuite, vous allez cliquer sur le bouton situé à droite de Envoyer (voir Figure 3.15). Une nouvelle fenêtre s'ouvre alors.



Figure 3.15
Envoyer une image.

Dans cette fenêtre, vous allez choisir la provenance de votre image :

- **Depuis votre ordinateur.** Choisissez une image sur votre ordinateur et transférez-la directement sur votre hébergement.
- **Faire un lien vers une adresse web.** Votre image est déjà disponible en ligne. Vous allez donc fournir son URL ainsi que plusieurs informations qui vous permettront de l'insérer dans votre article.
- **À partir de la bibliothèque de médias.** À force de mettre des médias en ligne, vous allez vous constituer une bibliothèque qui sera disponible directement à partir du troisième onglet de cette page.

Pour notre exemple, nous allons choisir de transférer une image directement depuis l'ordinateur. Vous allez donc cliquer sur le bouton Envoyer. Une fenêtre va apparaître pour que vous puissiez sélectionner l'image à transférer sur votre hébergement (voir Figure 3.16).

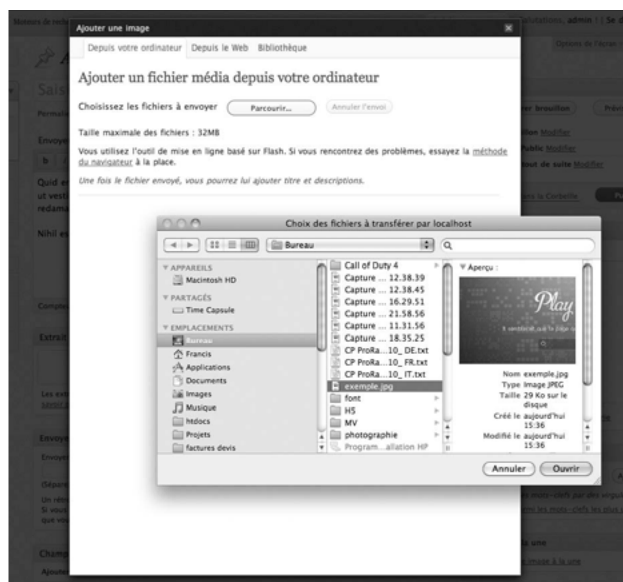


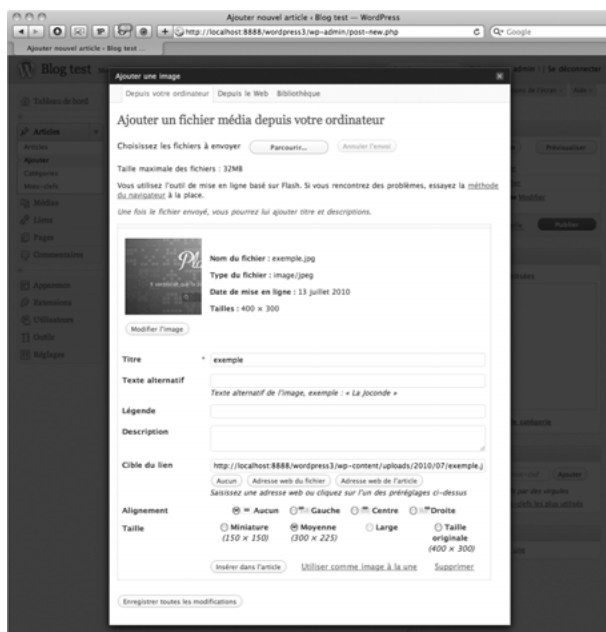
Figure 3.16
Choisir le fichier à transférer.

Une fois que vous avez trouvé cette image, validez votre choix ; l'image est directement transférée.

Dans la même fenêtre apparaissent alors une vignette de l'image ainsi que toute une série d'informations à renseigner (voir Figure 3.17).

Ici, vous allez pouvoir lui donner un titre, une légende, c'est-à-dire un titre alternatif, puis une description. Ces informations sont cependant facultatives, même s'il est recommandé d'indiquer au moins un titre descriptif.

Figure 3.17
Informations concernant le média à insérer.



En dessous, vous allez pouvoir récupérer l'URL du média que vous venez de charger sur le serveur de votre site. Ici, vous devez choisir si vous souhaitez que votre image soit associée à un lien et, si oui, à quel type de liens. En tout, vous avez trois possibilités :

- **Aucun(e).** Ici, pas de lien. Vous ne pouvez pas cliquer sur l'image dans l'article.
- **URL du fichier.** Qui est sélectionné par défaut. Cela signifie que, si vous cliquez sur l'image dans l'article, vous allez accéder à l'URL de l'image qui va apparaître seule dans une nouvelle fenêtre du navigateur. Pour revenir sur le blog, vous devrez utiliser le bouton Annuler ou Afficher la page précédente de votre navigateur.
- **URL de l'article.** Ici, en cliquant sur l'image, vous accédez à une page du blog qui est dédiée à votre image, où le visiteur aura la possibilité de laisser des commentaires. Cela peut se révéler très utile pour les artistes notamment, qui souhaiteraient utiliser WordPress comme portfolio.

Pour notre exemple, nous allons choisir l'option Aucun. L'image sera visible dans l'article, mais il ne sera pas possible de cliquer dessus pour aller sur une autre page.

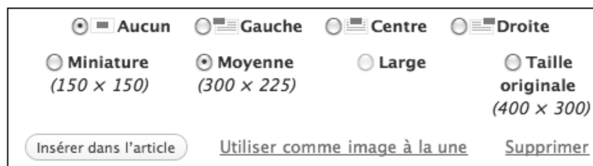
Vous devez ensuite choisir le positionnement de cette image dans le texte de l'article, ainsi que la taille de l'image à afficher : miniature, moyenne et taille originale. Pour notre exemple, vous allez aligner l'image à gauche et avec la taille originale.

Tout en bas, vous allez également trouver un lien qui se nomme Ajouter une vignette à l'article (voir Figure 3.18). En cliquant sur ce lien, vous allez pouvoir utiliser l'image que vous venez de charger comme vignette sur votre site. Cette fonctionnalité, apparue depuis la version 2.9 de WordPress, vous permet donc d'insérer des vignettes sur vos pages d'article,

sur la page d'accueil ou encore sur les pages d'archives. Cependant, tous les thèmes n'intègrent pas encore cette fonction. Vous pouvez tout de même la tester avec Twenty Ten, qui l'intègre. Nous verrons également au Chapitre 6 comment utiliser et insérer cette fonction dans votre thème WordPress.

Figure 3.18

Lien pour utiliser une image "à la une".



Une fois ces différentes informations saisies, cliquez sur le bouton Insérer dans l'article (notez ici la possibilité d'enregistrer toutes les modifications sans pour autant insérer l'image dans l'article).

Votre image apparaît alors dans votre article d'une manière différente, selon que vous utilisez l'éditeur visuel ou l'éditeur HTML.

Si vous utilisez l'éditeur visuel, vous allez pouvoir modifier certains aspects de l'image. Pour cela, vous allez cliquer sur votre image, puis sur le petit pictogramme faisant office d'image (voir Figure 3.19).

Figure 3.19

Fonctions supplémentaires pour l'insertion d'une image.



Une nouvelle fenêtre s'ouvre alors et vous permet de personnaliser encore un peu plus l'affichage de votre image. Ici, nous avons modifié la légende pour avoir un petit texte situé sous l'image.

Vous pouvez également modifier le positionnement et certains aspects du style de l'image en cliquant sur l'onglet Options avancées (voir Figure 3.20).

Enregistrez vos modifications. Vous pouvez alors visualiser votre article en cliquant sur le bouton Aperçu proposé en haut à droite du titre de l'article. Une nouvelle fenêtre s'ouvre et l'image apparaît bien insérée à droite comme prévu (voir Figure 3.21). Vous pouvez également passer la souris par-dessus l'image pour voir si elle n'est pas "cliquable".

Figure 3.20

Options avancées d'insertion d'une image.

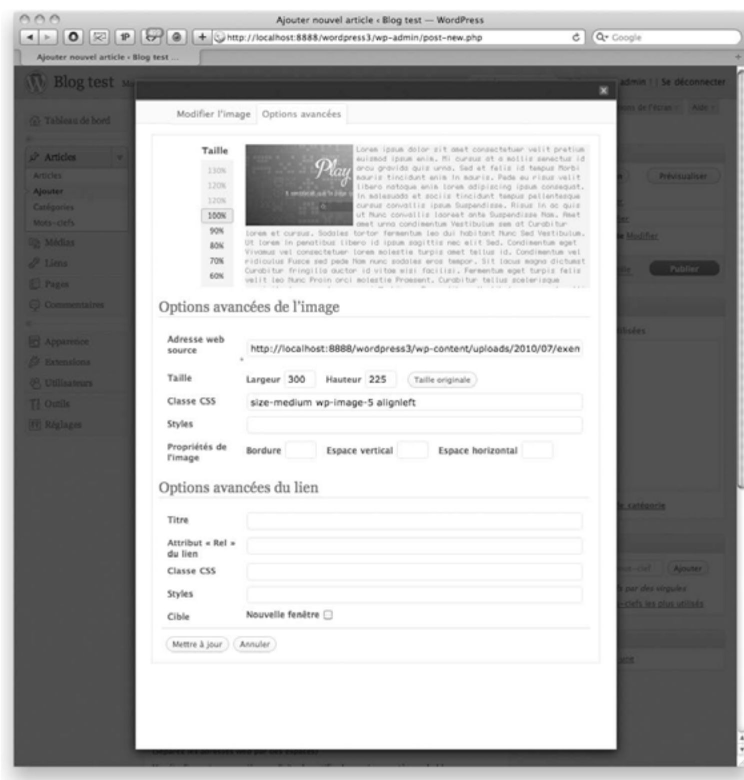


Figure 3.21

Image insérée dans l'article.



Insertion d'un autre type de média

Pour insérer un autre type de média en utilisant les autres boutons proposés, vous utiliserez la même méthode que pour une photo ou une image. Cependant, le résultat ne sera pas le même, et vous réaliserez rapidement que l'insertion d'une vidéo ou d'un fichier audio *via* la bibliothèque de médias n'est pas toujours optimale. Un grand nombre d'utilisateurs se tournent rapidement vers des extensions complètes et qui permettent d'insérer facilement

des vidéos en provenance de sites comme YouTube, DailyMotion, ou encore des vidéos hébergées sur leur serveur.

Il en est de même pour les fichiers audio. WordPress va jusqu'à proposer des extensions vous permettant de transformer votre blog en un véritable podcast. Pour plus d'informations, allez directement au Chapitre 4, dans la section réservée aux extensions.

Création d'une galerie

Quand vous cliquez sur le bouton Envoyer, la nouvelle fenêtre qui s'ouvre donne accès à trois onglets. Nous venons de voir le premier, et les deux autres sont nommés Galerie et Bibliothèque de médias. Si la bibliothèque de médias vous donne accès à tous les fichiers que vous avez déjà transférés sur votre hébergement, l'onglet Galerie va, comme son nom l'indique, créer une galerie dans votre article. Cela ne concerne donc que les photos et images.

Cela peut paraître logique, mais il vous faudra deux images pour créer une galerie. Une fois que vous aurez deux images disponibles, elles vont apparaître dans l'onglet Galerie (voir Figure 3.22).

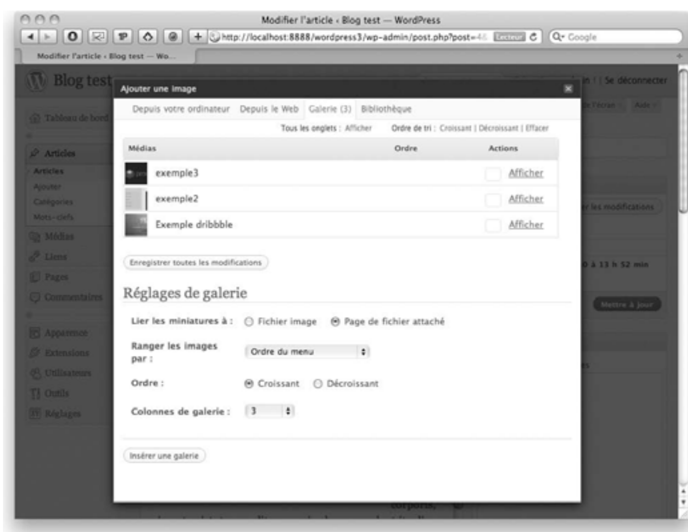


Figure 3.22
Galerie.

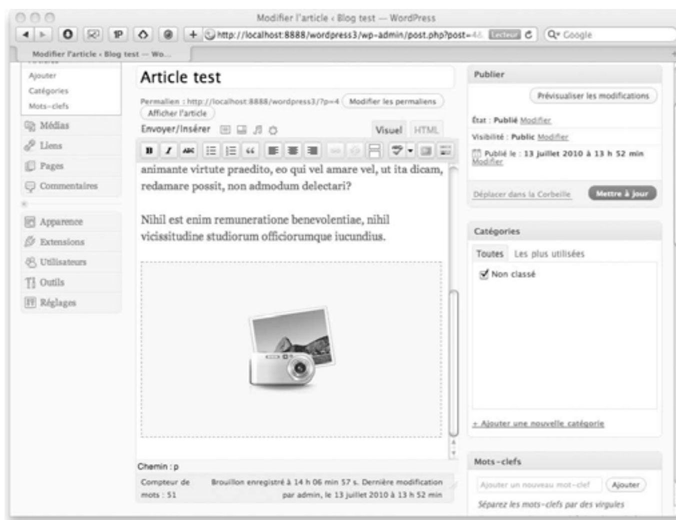
Sous les images à insérer, vous allez trouver les paramètres de la galerie à créer. Là, vous devez renseigner plusieurs informations :

- **Lier les miniatures à...** Ici, vous allez choisir si les vignettes de la galerie seront liées aux images directement ou si vous allez créer des "attachments" à cette image. En gros, vous avez la possibilité d'afficher l'image à sa taille normale en choisissant la première option ou, alors, vous pouvez insérer un visuel de ces images dans le design de votre site avec la possibilité pour les visiteurs de laisser des commentaires.
- **Ranger les images par...** Vous pouvez choisir l'ordre d'affichage des vignettes. Vous avez pour cela plusieurs possibilités : ordre du menu, titre, date et aléatoire.
- **Ordre.** Vous pouvez également choisir l'ordre d'affichage en décidant s'il doit être ascendant ou descendant.
- **Colonnes de galerie.** Choisissez ici le nombre de colonnes que doit faire votre galerie.

Une fois tous les renseignements remplis, vous pouvez créer votre galerie. Elle va apparaître sous forme différente selon que vous utilisez l'éditeur visuel (voir Figure 3.23) ou HTML (sous forme de shortcode `[gallery]`).

Figure 3.23

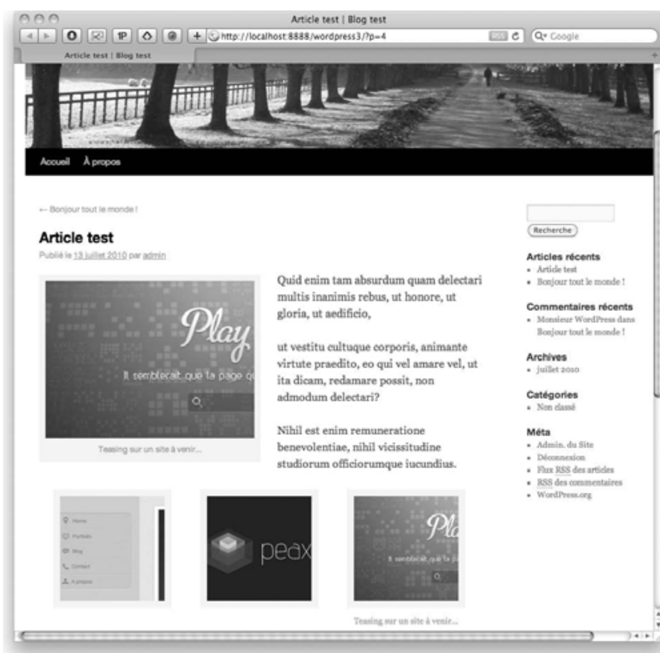
Affichage de la galerie dans l'article.



En visualisant votre article, vous découvrirez alors une galerie qui comprend les différentes images que vous avez insérées (voir Figure 3.24).

Figure 3.24

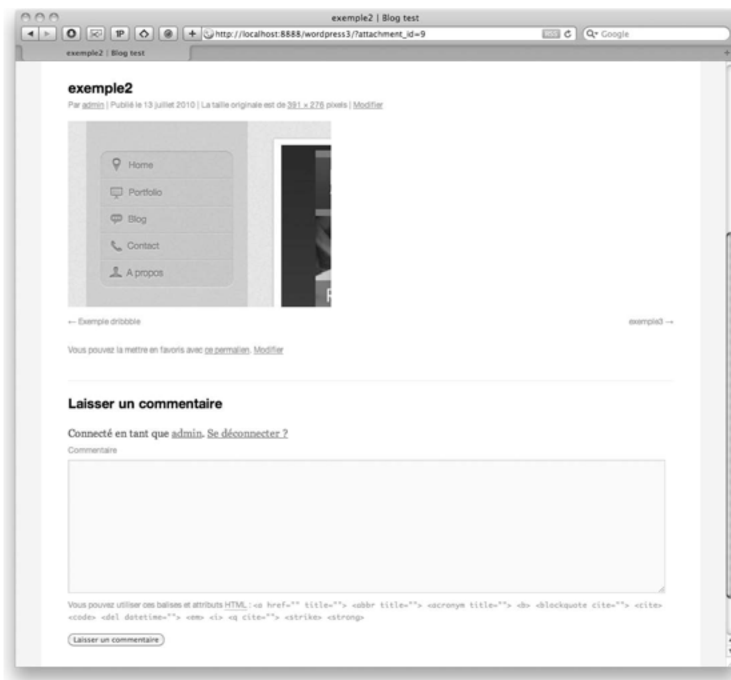
Galerie dans l'article.



En cliquant sur chacune d'elles, vous aurez accès au permalien de chaque image (voir Figure 3.25), donc à leur URL propre, et les visiteurs auront mê×me la possibilité de laisser un commentaire.

Figure 3.25

Permalien de média.

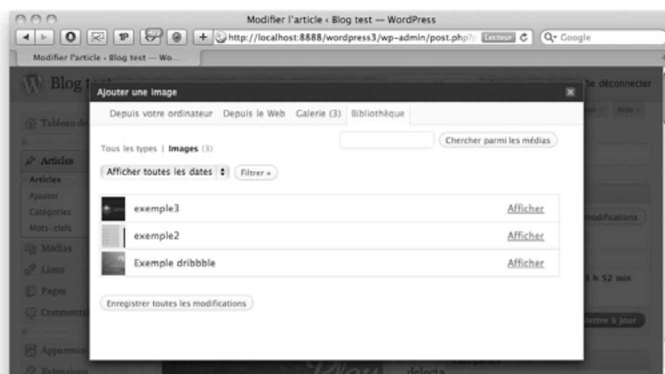


Insérer un média à partir de l'onglet Bibliothèque

La bibliothèque de médias a pour objectif de référencer tous les médias que vous avez pu transférer sur votre hébergement. Chacun de ces médias a une URL propre. Il peut arriver que vous souhaitiez réutiliser une photo qui a déjà servi pour un article précédent. Pour cela, vous devez aller dans la fenêtre Insérez un média et cliquer sur le troisième onglet Bibliothèque de médias (voir Figure 3.26).

Figure 3.26

Insérer un média à partir de la bibliothèque.



Choisissez l'image que vous souhaitez insérer dans votre article en cliquant sur le bouton Afficher. Les différentes informations concernant l'image apparaissent, et vous pouvez alors choisir la taille ainsi que le positionnement de votre image, comme nous l'avons vu précédemment. Validez le tout en cliquant sur le bouton Insérer dans votre article. L'image apparaît alors, comme convenu, dans votre article en cours de rédaction.

Changer l'emplacement de stockage des médias

Cet emplacement est défini par défaut au départ. Il s'agit du dossier uploads, qui se situe sous le dossier wp-content. Il est cependant modifiable dans les réglages de WordPress, sous l'onglet Réglages et le sous-menuMédias. Trouvez la ligne Envoi de fichiers et modifiez l'emplacement des fichiers. (voir Figure 3.27).

Figure 3.27
Emplacement des médias sur l'hébergement.

Notez qu'à partir de cette page vous pouvez aussi organiser votre dossier chronologiquement. Vos médias sont alors associés à des années et à des mois. Cela peut être très utile dès lors que vous gérez beaucoup d'images ou de vidéos par exemple.

Faites bien attention également à ce que l'emplacement choisi pour stocker vos médias ne soit pas directement dans le dossier de votre thème car, si vous supprimez celui-ci pour en changer, vos médias ne seraient plus visibles sur votre blog.

Taille des images

Les dimensions de la miniature et de la taille moyenne sont définies par défaut dès l'installation de WordPress, mais vous pouvez les modifier en fonction de vos besoins. Ces tailles "miniature" et "moyenne" sont paramétrables sous l'onglet Réglages, puis sous-menu Médias, directement sous la définition des liens par défaut pour un média.

Sur cette page, un bloc intitulé Taille des images va vous permettre de fixer des tailles pour les miniatures, la taille moyenne et la grande taille (voir Figure 3.28).

Figure 3.28
Taille des images.

Par défaut, celles-ci sont fixées à 150 × 150 pixels en ce qui concerne les miniatures, et 300 × 300 pixels pour la taille moyenne. À vous de fixer les tailles appropriées et de choisir enfin le type d'image que vous souhaitez voir apparaître dans votre article.

Intégration et taille des vidéos (affichages distants)

Depuis la version 2.9 de WordPress, vous avez la possibilité d'intégrer des vidéos en provenance de nombreux sites (YouTube, Vimeo, etc.) en donnant uniquement l'URL de cette vidéo. WordPress traduit directement cette URL pour en tirer la vidéo et l'afficher sur votre article.

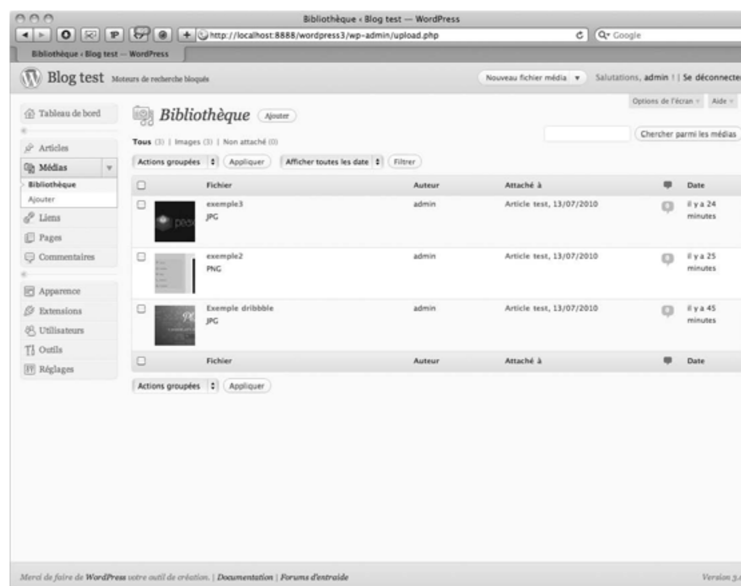
Sur la page Médias des réglages, vous avez la possibilité d'activer ou de désactiver cette fonction. En dessous, vous pouvez également choisir la taille des vidéos qui vont s'afficher dans vos articles (voir Figure 3.29).

Figure 3.29
Affichages distants.

Gestion de la bibliothèque de médias

Comme nous venons de le voir, la bibliothèque a pour but de référencer tous les médias utilisés sur le blog. Il vous est tout à fait possible de gérer cette bibliothèque *via* une page de l'interface d'administration de WordPress qui lui est dédiée. Elle se trouve sous l'onglet Gérer et dans le sous-menu Bibliothèque de médias (voir Figure 3.30).

Figure 3.30
Bibliothèque de médias.



Navigation dans la bibliothèque de médias

Si vous utilisez beaucoup de médias comme des images ou des vidéos, vous devrez rapidement faire un tri pour pouvoir trouver facilement ce que vous cherchez. Pour vous aider, vous

pouvez ici chercher par type de média, par date, ou encore *via* un formulaire de recherche (voir Figure 3.31).

Figure 3.31
Navigation dans les médias.

Pour chaque média, vous pourrez visualiser le ou les articles dans lesquels il est présent, ainsi qu'aller sur son permalien sur le blog.

Modification ou suppression d'un média

Pour modifier un média, il vous suffit de cliquer sur son nom dans la colonne Média, et vous arrivez directement sur une nouvelle page qui vous permet de modifier toutes les informations que vous avez saisies au moment de son ajout dans la bibliothèque.

Pour supprimer un média, il suffit de cocher la case située avant la vignette du média et de cliquer en haut de la page sur le bouton Supprimer.

Insérer une vidéo *via* l'éditeur visuel de WordPress

Nous avons vu précédemment qu'il était possible d'utiliser les fonctions des deux éditeurs pour ajouter des médias dans vos articles. L'éditeur visuel propose notamment d'insérer une vidéo (voir Figure 3.32). Pour cela, vous allez cliquer sur le bouton prévu à cet effet (voir au Tableau 3.01 les boutons de l'éditeur visuel).

Une nouvelle fenêtre s'ouvre. Elle comprend deux onglets : Général et Avancé.

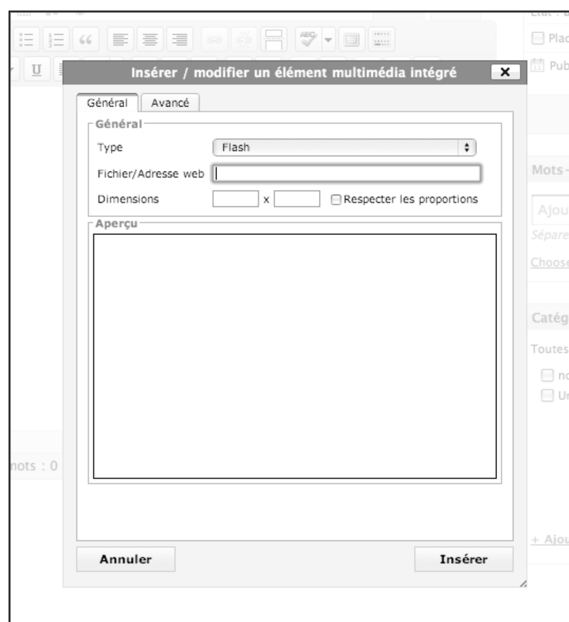


Figure 3.32
Insérer une vidéo avec l'éditeur visuel.

L'onglet Général vous demande de :

- définir le type de vidéo parmi ceux qui sont proposés (QuickTime, Shockwave, Real Media, Flash et Windows Media) ;
- saisir l'URL du fichier vidéo ;
- définir les dimensions voulues pour votre vidéo.

L'onglet Avancé, quant à lui, vous permet de définir d'autres données en fonction du type de fichier vidéo que vous aurez choisi (voir Figure 3.33).

Une fois que vous avez saisi toutes les informations nécessaires, cliquez sur le bouton Insérer et votre vidéo apparaît alors dans le corps du texte de votre article (voir Figure 3.34).

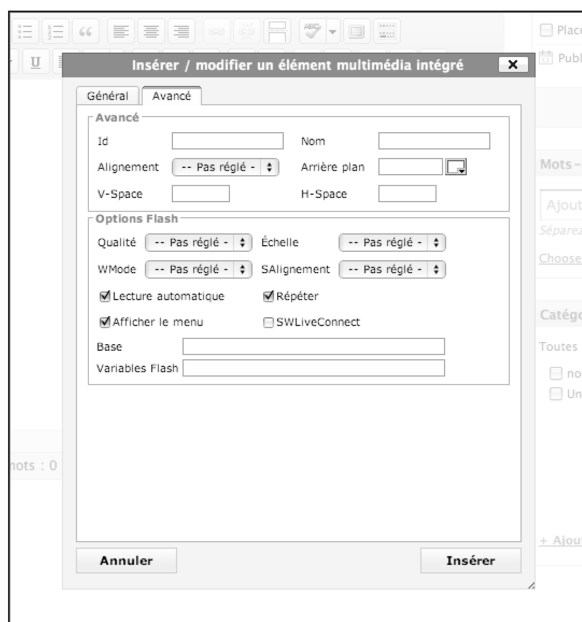


Figure 3.33
Éditeur visuel – Insertion de vidéo – Onglet Avancé.

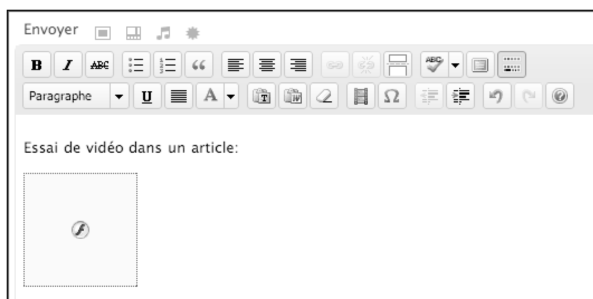


Figure 3.34
Vidéo dans l'article sur l'éditeur visuel.

Options avancées de rédaction d'un article

Directement sous l'éditeur de texte, vous avez différents blocs qui vous permettent de renseigner des informations concernant votre article et notamment son affichage. Aucune de ces options n'est obligatoire. Cependant, en fonction de vos besoins, elles pourront se montrer très utiles.

Dans cette section, nous ne les passerons pas toutes en détail. Certaines options concernent principalement l'apparence des articles sur le thème et seront donc détaillées plus longuement dans la partie de ce livre consacrée à la création d'un thème pour WordPress.

Extrait

Figure 3.35
Extrait d'un article.

Extrait

Les extraits sont des résumés facultatifs de vos articles, écrits à la main. Vous pouvez les utiliser dans votre thème

Au moment de la présentation des deux éditeurs de WordPress, nous avons fait la connaissance de la fonction More, qui permet de ne proposer qu'un morceau de l'article sur la page d'accueil du blog (voir Figure 3.35).

Ici, vous avez la possibilité d'écrire un résumé de votre article plutôt qu'en proposer les premières lignes. C'est souvent plus pertinent pour amener un sujet intéressant. Cela peut servir de *teaser*.

Si vous le souhaitez, vous pouvez écrire votre résumé dans l'espace qui lui est alloué, et ce morceau de texte apparaîtra sur la page d'accueil pour inciter le lecteur à aller lire l'article sur son permalien.

Cependant, comme pour la fonction More, vous devrez vérifier si votre thème vous permet d'afficher cet extrait. Ce paramétrage vous est expliqué en détail au Chapitre 6.

Rétroliens et pings

Figure 3.36
Rétroliens.

Envoyer des rétroliens

Envoyer un rétrolien vers :

(Séparez les adresses web par des espaces)

Un rétrolien est une manière explicite de notifier les anciens systèmes de blog que vous avez fait un lien vers eux. Si vous faites un lien vers des sites WordPress, ils seront notifiés automatiquement à l'aide des pings (en), sans que vous n'ayez rien à faire.

Les rétroliens permettent de notifier à d'autres blogs que vous avez créé un lien vers eux (voir Figure 3.36). Cette fonction est aujourd'hui dépréciée car les systèmes de blogs gèrent cette fonction de manière automatique. Cependant, elle existe et elle vous permet de saisir une ou plusieurs URL que vous souhaiteriez notifier, le cas échéant.

Champs personnalisés

Figure 3.37
Les champs personnalisés.

Champs personnalisés

Nom	Valeur

Ajouter un nouveau champ personnalisé :

Nom	Valeur

Ajouter un champ personnalisé

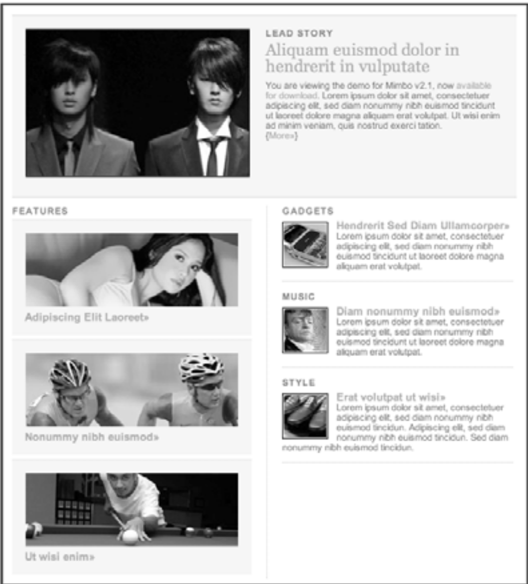
Les champs personnalisés peuvent être utilisés afin d'ajouter des données supplémentaires à vos articles. Vous pouvez les utiliser dans votre thème (en).

Par défaut, les champs personnalisés (voir Figure 3.37) vous permettent d'ajouter quelques informations supplémentaires au niveau de votre article, concernant par exemple votre humeur, ce que vous lisez en ce moment ou encore un texte qui résumerait en trois mots le contenu de l'article :

- Humeur : en pleine forme !! ;
- En train de lire : *Des souris et des hommes*, de Steinbeck ;
- Politique étrangère.

Les champs personnalisés sont également très utilisés ces derniers temps sur les thèmes magazines pour faire apparaître des images sur la page d'accueil. Vous définissez l'image au niveau du champ personnalisé et elle apparaît sur la page d'accueil, en face de l'article (voir Figure 3.38).

Figure 3.38
Utilisation des champs personnalisés pour afficher des images.



Leur paramétrage est expliqué en détail au Chapitre 6.

Figure 3.39
Discussion.

Discussion
☑ Autoriser les commentaires.
☑ Autoriser les rétroliens et pings sur cette page.

Vous allez décider ici si vous souhaitez autoriser ou non les commentaires pour votre article (voir Figure 3.39). Idem pour les rétroliens.

Commentaires

Si votre article a des commentaires, ils viendront s'afficher ici. À partir de cette page, vous pourrez d'ailleurs les gérer et envoyer une réponse (voir Figure 3.40).

Figure 3.40
Commentaires.

Commentaires

Auteur	Commentaire
Monsieur WordPress wordpress.org/	13/07/2010 à 9 h 51 min Bonjour, ceci est un commentaire. Pour supprimer un commentaire, connectez-vous, et affichez les commentaires de cet article. Vous pourrez alors les modifier ou les supprimer.

Auteur

Choisissez ici l'auteur de l'article. Il est modifiable, même une fois que l'article est publié (voir Figure 3.41).

Figure 3.41
Auteur.

Auteur
admin

Identifiant de l'article

Par défaut, WordPress va donner un identifiant à votre article (voir Figure 3.42). Cet identifiant dépendra de la structure des permaliens donnée à votre blog (voir plus loin dans ce chapitre). Pour faire simple, disons que WordPress va attribuer une URL par défaut à votre article.En haut de la page, sous le titre, vous pouvez changer le permalien de l'article. Si la structure des permaliens est celle définie par défaut, vous ne pourrez pas effectuer de modifications. Cependant, WordPress vous donne un lien vers la page qui vous permettra de modifier cette structure (voir Figure 3.42).

Figure 3.42
Identifiant de l'article.

Article test
Permalien : http://localhost:8888/wordpress3/?p=4 Modifier les permaliens Afficher l'article

Si vous choisissez par exemple une structure qui va afficher le nom de votre article, vous pourrez alors modifier les mots inclus dans cette URL et les remplacer par exemple par des mots-clefs.

WordPress vous donne également la possibilité de récupérer l'URL de base de vos articles, qui peut faire office d'URL raccourcie étant donné sa taille réduite.

État de publication

La page de rédaction d'un article compte une colonne latérale, à droite, qui vous permet de définir différentes informations concernant le statut de l'article, mais aussi de définir des mots-clefs et une ou plusieurs catégories pour votre article. Le premier bloc est destiné à tout ce qui touche à la publication d'un article (voir Figure 3.43).



Figure 3.43
Bloc Publier de la colonne latérale.

Prévisualisation et enregistrement de l'article

Tout d'abord, vous avez un bouton qui vous permet de prévisualiser votre article dans une nouvelle fenêtre. L'URL de l'article n'est pas disponible pour les visiteurs de votre blog. Elle est uniquement visible par vous, dès lors que vous souhaitez voir à quoi va ressembler l'article que vous êtes en train de rédiger.

À gauche, vous avez un deuxième bouton qui vous permet d'enregistrer votre article. Son enregistrement n'entraîne pas ici de modification de son état tant que vous ne le modifiez pas sur la ligne juste en dessous. Là, si vous changez son état, celui-ci ne sera pris en compte qu'après l'enregistrement.

États de l'article

Sous WordPress, il existe différents états pour l'article :

- **Publié.** Article publié et visible par tous.
- **Brouillon.** Enregistré mais pas publié.
- **Programmé.** Planifié pour une publication dans un futur proche.
- **En attente de relecture.** Doit être validé par un administrateur avant d'être publié.
- **Privé.** Visible uniquement par les utilisateurs enregistrés du blog.

Tant que votre article n'est pas enregistré, il aura le statut de "brouillon".

Visibilité de l'article

WordPress vous permet d'avoir trois types de visibilité pour votre article (voir Figure 3.44a). Il peut être public, donc visible par tout le monde. Il peut être protégé par un mot de passe ; dans ce cas, vous devez fournir le mot de passe pour que les visiteurs puissent le visualiser. Enfin, il peut être privé, ce qui veut dire qu'il ne sera visible que par les visiteurs qui ont un compte d'accès au blog, *via* un rôle ; bien souvent, celui utilisé est "abonné".

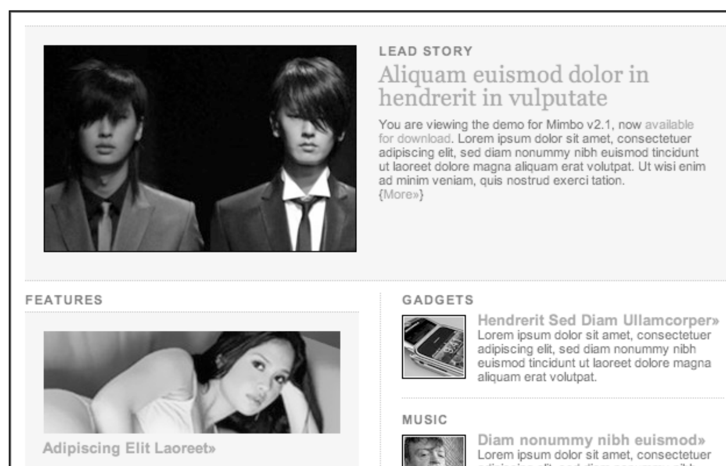
Figure 3.44a
Visibilité de l'article.



Placer un article en tête de la page d'accueil

Depuis la version 2.7 de WordPress, vous avez la possibilité de mettre en avant un article sur la page d'accueil de votre blog. Celui-ci s'affichera tout en haut de la page, au-dessus des autres, passant ainsi au-dessus de la suite antéchronologique d'un blog. Pas mal de blogs en font souvent une section particulière appelée À la une (voir Figure 3.44b).

Figure 3.44b
Article À la une.



Cette option peut servir pour mettre en lumière des articles qui intéresseraient éventuellement les lecteurs de votre blog et qui seraient probablement perdus dans la masse de billets si cette option n'existait pas.

Choisir la date de publication d'un article

Par défaut, votre article est publié immédiatement, mais dans WordPress, vous avez la possibilité de choisir la date et l'heure auxquelles votre article sera mis en ligne (voir Figure 3.45). Pour cela, il vous suffit de cliquer sur le lien Modifier, à droite du petit calendrier et du texte Publier tout de suite.

Figure 3.45
Changer la date de publication d'un article.

Cette option peut être intéressante dans bien des cas, par exemple si vous partez en vacances mais que vous souhaitez tout de même publier des articles pour vos lecteurs. Ainsi, vous rédigez vos articles et vous pouvez choisir des dates réparties sur les jours et semaines qui suivent. Une fois le billet planifié, cliquez sur Publier et celui-ci passera en statut Programmé. Il n'apparaîtra sur le blog qu'à la date que vous aurez choisie.

Options de l'écran

Comme sur le tableau de bord, vous avez ici un panneau vous permettant de choisir les blocs que vous souhaitez voir apparaître quand vous rédigez un article (voir Figure 3.46).

Figure 3.46
Options de l'écran de rédaction d'un article.

Vous pouvez également choisir d'afficher la page sur une ou deux colonnes.

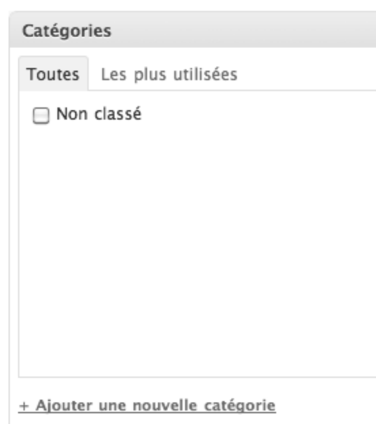
Découverte des catégories

Insertion des catégories dans un article

Directement sous les tags, vous allez attribuer au moins une catégorie à votre article. Celles-ci permettent de classer les articles par sujet. Nous sommes dans le même domaine que les mots-clefs, sauf qu'ici nous éviterons de classer nos articles dans beaucoup de catégories à

la fois. La catégorie sera le sujet de l'article, alors que les mots-clefs seront plutôt les mots qui caractérisent le même article (voir Figure 3.47).

Figure 3.47
Gestion des catégories dans la page de création d'un article.



Il faut également savoir qu'un article doit être associé au minimum à une catégorie, alors que vous pouvez très bien ne pas proposer de mots-clefs.

Tout comme les mots-clefs, les catégories facilitent beaucoup la navigation pour les visiteurs. Bien souvent, on affichera la liste des catégories dans la colonne latérale, pour permettre à l'internaute de trouver facilement le sujet qu'il recherche (voir Figure 3.48).

Figure 3.48
Affichage des catégories dans la colonne latérale.



Ici aussi, essayez de limiter le nombre de catégories. Beaucoup de personnes conseillent de n'en choisir qu'une par article pour simplifier au maximum leur gestion.

Il n'existe par défaut qu'une seule catégorie, qui s'appelle Non classé. Si vous écrivez votre premier article, c'est la seule qui vous sera proposée. Vous allez donc probablement devoir en créer une ou plusieurs autres pour coller au mieux avec le contenu de votre article.

Pour cela, cliquez sur le lien + Ajouter une nouvelle catégorie dans le bloc consacré aux catégories. Une nouvelle ligne apparaît. Saisissez le nom de votre nouvelle catégorie et choisissez ou non une catégorie parente. Si c'est le cas, votre nouvelle catégorie deviendra une sous-catégorie. Mais, si vous créez votre première catégorie, vous ne choisirez pas de catégorie parente (voir Figure 3.49).

Figure 3.49
Création d'une catégorie pour un article.

Vous pouvez également choisir ici d'afficher les catégories de deux manières : soit vous affichez "toutes les catégories", soit vous n'affichez que les catégories les "plus utilisées".

Gestion des catégories

Comme pour les mots-clefs, il existe une page où vous allez gérer l'ensemble de vos catégories. Cette page se situe sous l'onglet Articles et le sous-menu Catégories. Sur cette page, vous allez pouvoir créer, modifier et supprimer une ou plusieurs catégories.

Création, modification et suppression d'une catégorie

À l'instar de la page de gestion des mots-clefs, la page de gestion des catégories est divisée en deux colonnes : celle de gauche, qui vous permet de créer une nouvelle catégorie, et celle de droite, qui vous permet de modifier les catégories existantes (voir Figure 3.50).

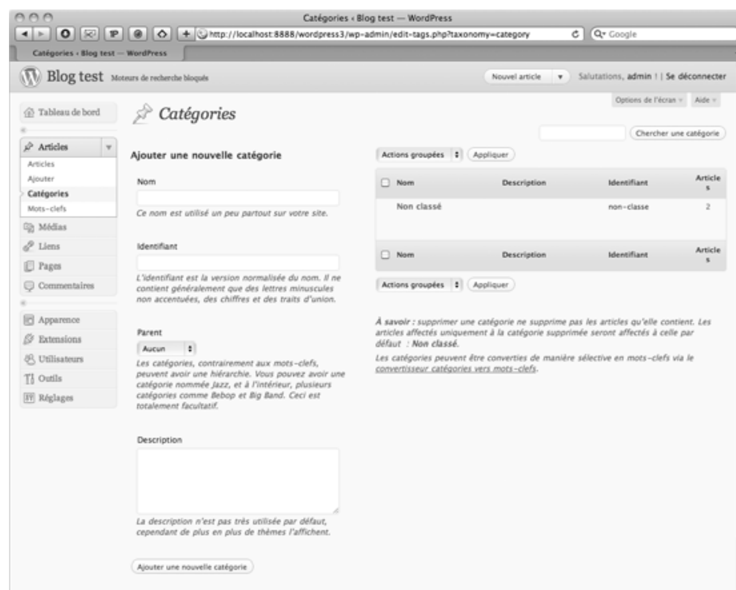


Figure 3.50
Page de gestion des catégories.

Pour créer une nouvelle catégorie, vous devez renseigner différentes informations :

- **Nom de la catégorie.**
- **Identifiant de la catégorie.** Important parce que c'est lui qui va apparaître dans l'URL de la page des catégories, si vous choisissez la structure de cette URL.
- **Catégorie mère.**
- **Description.** Elle peut être utile car certains thèmes ont tendance à l'afficher sur la page des catégories.

Pour modifier ou supprimer une catégorie, le fonctionnement est identique que pour les mots-clefs. Vous remarquerez cependant que la catégorie par défaut Non classé n'est pas supprimable.

Découverte des mots-clefs

Directement sous le bloc Publier se trouve un autre bloc appelé Mots clefs. Dans cette section, vous allez apprendre à choisir et insérer des mots-clefs dans votre article, mais nous en profiterons aussi pour apprendre à les gérer, *via* la page de l'interface d'administration qui leur est dédiée.

Insertion des mots-clefs dans un article

Rédiger un article ne consiste pas uniquement à créer un contenu et à trouver un titre. Cet article, vous allez devoir l'associer à différentes informations qui pourront être très utiles aux visiteurs de votre blog. Les mots-clefs en font partie.

Qu’est-ce qu’un mot-clef ? En anglais, on les appelle des "tags". Ce sont des mots qui caractérisent votre article. Ils sont très prisés et utilisés sur le Web de nos jours. Ils pourront s’afficher au niveau de l’article (voir Figure 3.51) mais aussi au niveau de la colonne latérale du blog, *via* notamment un widget qui servira ainsi d’outil de recherche pour le visiteur (voir Figure 3.52). Ce widget va afficher tous les mots-clefs créés dans tous les articles sous forme de nuage. Plus un mot-clef est utilisé, plus il va être gros. C’est le principe même du nuage de tags. Il est donc important de ne pas créer trop de mots-clefs par article pour que ce nuage reste utilisable. Si vous avez en tout 500 mots-clefs par exemple sur votre blog, vous ne pourrez probablement pas y afficher le nuage. Essayez de vous limiter à une dizaine au maximum par article.

Figure 3.51
Présentation des tags au niveau d'un article.

Ce plan d’action mis en place sera amorti sur une petite dizaine d’années. Celui-ci me semble être une bonne initiative du fait que l’on prend en compte l’économie de l’énergie, la manière de consommer avant de penser à la production d’énergie renouvelable. Une initiative à encourager, donc, quand on sait qu’en plus de cela, Perpignan se situe dans la région la plus ensoleillée de France.

Tags: Ecologie, énergie positive, perpignan, ville écologique
Publié dans A ne pas manquer, Energie | Modifier | Aucun commentaire »

Figure 3.52
Présentation des tags dans un widget.



En cliquant sur un mot-clef sur le blog, vous accédez à une nouvelle page du blog qui référencera tous les articles qui ont été tagués avec le mot que vous avez choisi (voir Figure 3.53).

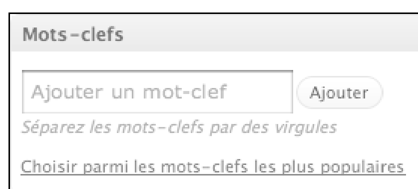
Dans la colonne latérale de votre page de rédaction d’article, vous avez donc un bloc où vous allez saisir les différents mots-clefs associés à votre article. Si leur utilisation est fortement conseillée, elle n’est cependant pas obligatoire, contrairement aux catégories que nous allons voir ensuite (voir Figure 3.54).

Essayez de trouver les mots-clefs qui caractérisent le mieux le message que vous voulez faire passer dans votre article.

Figure 3.53
Page d'un tag sur le blog.



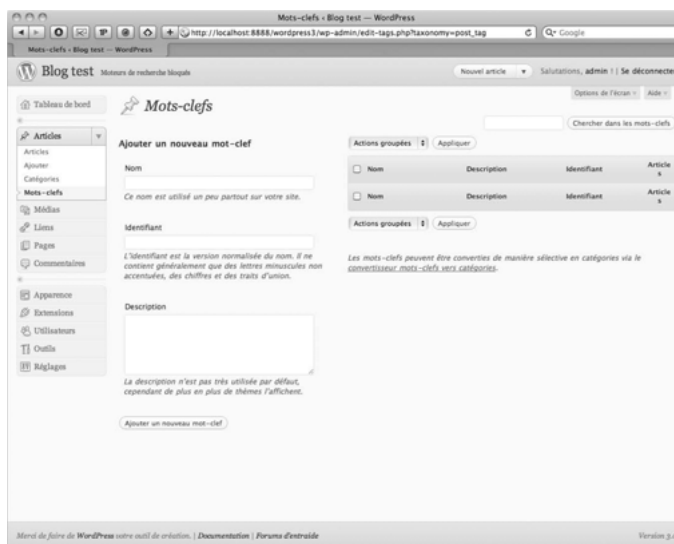
Figure 3.54
Les tags dans la rédaction d'un article.



Gestion des mots-clefs

Chaque tag créé est géré par WordPress. Sur l'interface d'administration, une page est dédiée à leur gestion (voir Figure 3.55). Elle se situe dans l'onglet Articles et le sous-menu Mots clefs. Sur cette page, vous pourrez aussi bien créer, modifier ou encore supprimer des tags.

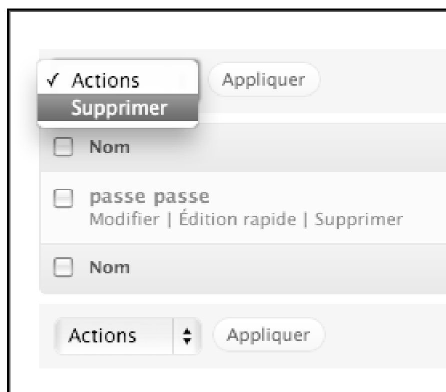
Figure 3.55
Page de gestion des tags.



La page est divisée en deux colonnes. Celle de gauche vous permet de créer un nouveau mot-clef en lui donnant un nom et un identifiant. La colonne de droite vous donne la possibilité de modifier les mots-clefs existants.

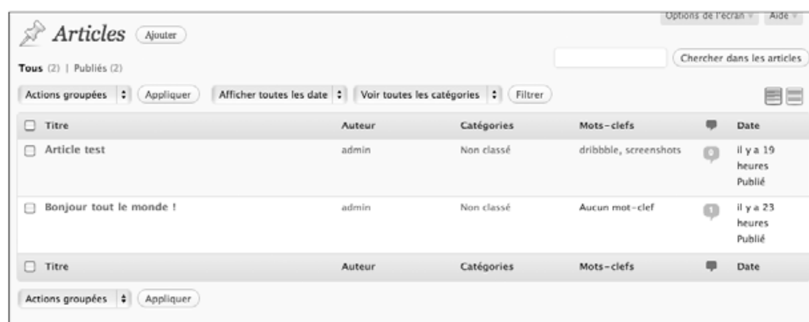
Ici, vous avez trois possibilités : Modifier, Édition rapide, qui correspond à une modification mais sous une forme différente, et Supprimer. Pour modifier le mot-clef, vous pouvez cliquer directement dessus, et pour le supprimer, vous pouvez cocher la case devant le mot-clef et choisir Supprimer dans le menu déroulant des actions disponibles (voir Figure 3.56).

Figure 3.56
Suppression d'un mot-clef.



Enfin, cette page vous permet de connaître également l'ensemble des articles comprenant un tag. Dans le tableau, vous avez à droite une colonne qui s'intitule Articles et qui référence le nombre d'articles contenant le mot-clef. Cliquez sur ce chiffre et vous aurez la liste des articles en question (voir Figure 3.57).

Figure 3.57
Nombre d'articles possédant ce mot-clef.



Insertion d'une vignette dans l'article

Depuis la version 2.9 de WordPress, vous avez la possibilité d'insérer des vignettes d'images dans vos articles (voir Figure 3.58). Cette vignette accompagnera votre article partout où elle pourra être affichée. Cette utilisation est très courante notamment sur les thèmes "magazines". Dans Twenty Ten, le nouveau thème par défaut de WordPress, la vignette est utilisée de manière assez originale puisqu'elle va remplacer l'image qui est utilisée dans l'en-tête du thème.

Figure 3.58

Insertion d'une image à la une.



Le processus est identique à ce que nous avons vu précédemment pour insérer une image directement dans un article. Une fois votre image chargée, elle apparaît sur la page de rédaction de votre article.

Au Chapitre 6, nous verrons comment insérer et utiliser cette fonction dans n'importe quel thème.

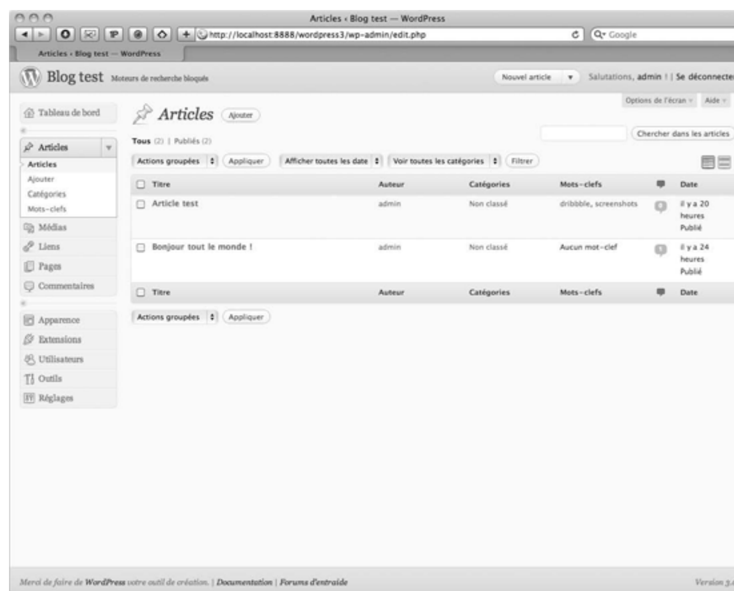
Gérer vos articles

Il vous arrivera peut-être de vouloir modifier des articles, même lorsqu'ils sont publiés, voire en supprimer certains. Pour cela, vous devez aller sur la page de gestion des articles qui est située sous l'onglet Articles et le sous-menu Modifier (voir Figure 3.59).

Le système de gestion des articles reprend le même concept que celui que nous avons vu précédemment pour les mots-clefs et les catégories.

Figure 3.59

Page de gestion des articles.



Navigation parmi les articles

Pour trouver un article, utilisez la fonction de recherche de cette page, située en haut à droite. C'est une fonction classique mais souvent sous-utilisée. Grâce à elle, vous trouverez rapidement ce que vous cherchez (voir Figure 3.60).

Figure 3.60
Formulaire de recherche de la page de gestion des articles.

Chercher dans les articles

Vous pouvez également effectuer des sélections en triant :

- **Par type d'article.** Tous les articles, publiés, brouillons ou encore par articles privés.
- **Par date.** WordPress propose un menu déroulant pour effectuer un tri dans les archives. Ce tri se fait par mois.
- **Par catégorie.** Vous avez la possibilité ici de trier l'ensemble de vos articles par catégorie.
- **Par auteur.** Dans le tableau regroupant l'ensemble des articles, vous trouverez une colonne Auteur. En cliquant sur un des auteurs présents, vous pourrez voir l'ensemble des articles qu'il a pu rédiger.
- **Par tag.** Vous pouvez aussi faire un tri par tag, au niveau de la colonne Tags dans le tableau des articles.

Modification et suppression d'un article

Pour modifier un article, il vous suffit de cliquer sur son titre ou d'utiliser les liens situés en dessous et qui vous permettront de créer l'action voulue. Vous pouvez également cocher la case devant la ligne de l'article et choisir une action dans le menu déroulant au-dessus du tableau.

En savoir plus sur les articles

À partir de la page de gestion des articles, vous avez la possibilité de voir les commentaires pour chaque article, en cliquant sur la "bulle" avec un chiffre qui correspond au nombre de commentaires pour l'article. La nouvelle page qui s'ouvrira sera la page de gestion des commentaires (voir Figure 3.61).



Figure 3.61
Accéder aux commentaires à partir de la page de gestion des articles.

Écrire et gérer une page

Différence entre page et article

Une page est considérée par WordPress comme un élément statique du blog, c'est-à-dire qui est en dehors du flux chronologique. Vous allez notamment l'utiliser pour créer des éléments distincts du blog, comme une page d'avant-propos, d'archives ou encore un formulaire de contact. Les articles, eux, sont des éléments dits dynamiques ; ils alimentent le contenu du

site et sont de ce fait très prisés par les moteurs de recherche car ils apportent du contenu frais sur le Web.

Il est donc important de bien comprendre la différence entre une page d'article du blog, autrement appelée permalien, et une page statique du blog dont nous allons parler dans la section suivante.

Création, modification et suppression d'une page

Pour créer une nouvelle page dans votre blog, allez sous l'onglet Page. Cliquez sur Ajouter. Vous arrivez directement sur la page qui va vous permettre de créer une nouvelle page pour votre blog (voir Figure 3.62). Là, vous retrouvez une grande partie de ce qui existe déjà pour la rédaction des articles.

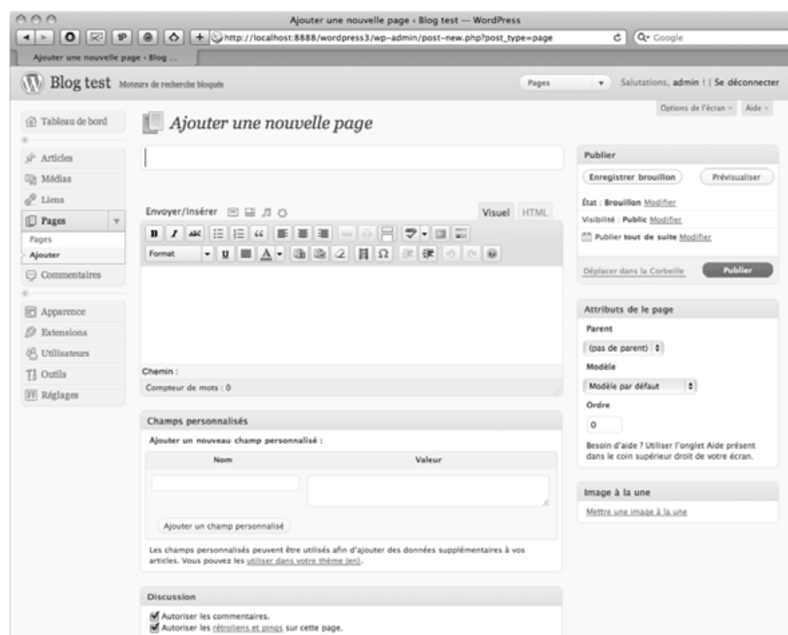


Figure 3.62
Création d'une nouvelle page.

Comme pour la rédaction d'un article, vous avez différentes options telles que les champs personnalisés, la gestion des commentaires et des pings, ainsi que la possibilité de choisir l'auteur de la page.

Mais la création de page a également ses propres spécificités :

- **Parent.** Vous avez la possibilité de créer des "sous-pages", de donner une hiérarchie à vos pages.
- **Modèle de page.** Vous pouvez assigner à votre page un modèle spécifique que vous aurez créé ou qui est fourni avec le thème. La définition de modèle se rapproche du

concept de template de WordPress. Ici, le contenu comme le design peuvent être déterminés de manière unique. La notion de modèle de page sera abordée de manière plus détaillée au Chapitre 6.

- **Ordre.** Les pages sont classées par ordre chronologique ; vous avez la possibilité de les classer dans un ordre arbitraire.

La notion d'état de publication, quant à elle, est la même que pour la rédaction d'un article.

À noter que vous pouvez également associer une page à une image à la une.

Gestion des pages

Pour gérer les différentes pages du blog, vous allez rester dans le même onglet ouvert pour la création d'une page du blog et vous allez cliquer sur le lien Modifier. Cette page est semblable à ce que vous avez déjà vu pour les articles, mais aussi les mots-clefs et les catégories, ou encore l'insertion d'une vignette :

- recherche *via* le formulaire ;
- tri par type de page ou par auteur ;
- modification des pages en cliquant sur leur titre ;
- modification ou suppression des pages *via* les liens Modifier, Édition rapide, Supprimer et Afficher ou Aperçu selon l'état de la page ;
- suppression ou modification en cochant la case devant chaque page et en choisissant l'action souhaitée dans le menu déroulant situé au-dessus du tableau ;
- accès aux commentaires et aux pings les concernant *via* la petite "bulle" prévue à cet effet.

Écrire et gérer les liens et les catégories de liens

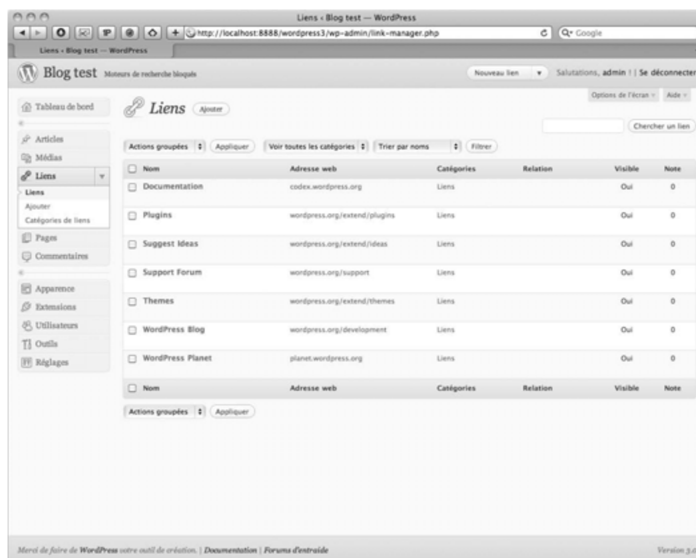
Définition

Lorsque vous installez WordPress, votre blog possède automatiquement ce qu'on appelle une liste de liens , pointant vers différents sites de WordPress (voir Figure 3.63).

La blogroll est une liste de liens que vous partagez avec vos lecteurs, un peu comme des listes de blogs d'amis que l'on voit souvent affichées sur différents sites.

Figure 3.63

Liste de liens WordPress.



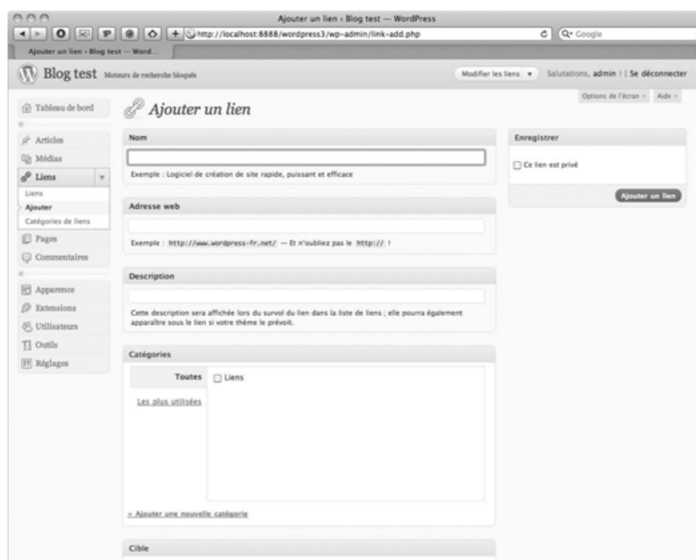
Création, modification et suppression d'un lien et d'une catégorie de liens

Création des liens

Vous commencerez par créer un premier lien. Vous allez également en profiter pour créer une catégorie de liens. Pour cela, allez sous l'onglet Liens, puis cliquez sur Ajouter ; vous arrivez directement sur la page de création d'un lien (voir Figure 3.64).

Figure 3.64

Page de création d'un lien.



Là, vous allez renseigner différentes informations :

- **Nom.** Indiquez le nom qui va apparaître sur la page web.
- **Adresse web.** Écrivez ici l'URL du site ou du blog pour lequel vous souhaitez créer un lien.
- **Description (facultatif).** Fournissez une courte description qui s'affichera au survol du lien par la souris.
- **Catégories.** C'est ici que vous allez associer vos liens avec une catégorie de liens. La seule catégorie proposée lorsque vous créez votre blog est la blogroll. Nous allons donc en créer une seconde que nous appellerons "blogs amis". Pour cela, cliquez sur le lien + Ajouter une nouvelle catégorie.

  Vous pourrez alors enregistrer votre nouvelle catégorie de liens en lui donnant tout simplement un nom.

  Notez que vous avez la possibilité de mettre le même lien dans plusieurs catégories.

Ensuite, WordPress vous propose plus d'options avancées, qui sont cependant facultatives et que vous pourrez afficher ou masquer grâce aux options de l'écran.

Tout d'abord, vous allez choisir la cible de vos liens, c'est-à-dire que vous déterminerez comment vont s'ouvrir les liens dès lors qu'ils seront cliqués. Vous avez trois possibilités :

- **_blank.** Ouverture du lien dans une nouvelle fenêtre.
- **_top.** Ouverture du lien dans un nouvel onglet.
- **Aucune.** Ouverture du lien dans la même fenêtre.

Puis WordPress vous propose de fournir des informations concernant la relation avec le propriétaire du site lié (XFN). Cette information, stockée dans le code sous la fonction `rel`, permet d'avoir des précisions sur votre "relation" avec la personne qui possède le site que vous allez lier ici (voir Figure 3.65). Ces informations sont cependant optionnelles et peu utilisées.

Figure 3.65

Relation avec le propriétaire du site lié (XFN).

Enfin, WordPress vous propose toute une série d'options pour l'affichage de vos liens :

- **Adresse de l'image.** Vous pouvez remplacer votre lien texte par une image. Il vous suffit de fournir son URL.
- **Adresse du flux RSS.** Vous pouvez fournir l'adresse du flux RSS du blog que vous liez.
- **Commentaires.** Laissez ici tout commentaire concernant ce lien.
- **Classement.** Les liens sont des listes créées par ordre alphabétique. Vous pouvez ici leur donner un ordre arbitraire.

Dans la colonne de droite, vous avez la possibilité de déterminer si le lien est privé ou pas. S'il est privé, le lien n'apparaîtra que pour les utilisateurs enregistrés du blog.

Options avancées

WordPress vous offre la possibilité d'associer quelques informations supplémentaires à votre lien :

- l'associer à une image ;
- l'associer à un flux RSS ;
- l'associer à des notes ;
- l'associer à un système de notation qui va de 0 à 10.

Gestion des liens

Tout se passe sur la page Modifier de l'onglet Liens. Nous reprenons ici les mêmes éléments de gestion que ceux que nous avons vus précédemment sur les autres pages de gestion, à savoir :

- possibilité de faire une recherche *via* le formulaire en haut à droite ;
- tri des liens par catégorie, par identifiant du lien, par nom, par adresse, par notes ;
- modification de lien en cliquant sur son nom ;
- modification ou suppression *via* les liens Modifier, Édition rapide, Supprimer et Afficher ou Aperçu selon l'état de la page ;
- suppression ou modification en cochant la case devant chaque lien et en choisissant l'action souhaitée dans le menu déroulant situé au-dessus du tableau ;
- accès direct sur le site lié en cliquant sur l'URL ;
- et, enfin, suppression de lien en cochant la case qui précède son nom et en cliquant ensuite sur le bouton Supprimer.

Gestion d'une catégorie de liens

Nous avons généré une nouvelle catégorie de liens directement en créant un lien. Sur WordPress, il existe une page dédiée à ces catégories de liens. Cette page se trouve toujours sous l'onglet Liens et sur la page Catégories de liens.

Sur cette page, vous avez la possibilité de créer et de gérer les différentes catégories de liens de la même manière que ce que nous avons pu faire précédemment. Les différentes informations que vous pourrez renseigner pour chaque catégorie de liens sont :

- **Nom de la catégorie.** Vous allez lui donner le nom de votre liste, "blogs amis".
- **Identifiant.** Très utile pour le référencement et avec l'utilisation de certaines URL. Il ne doit pas contenir d'accents.
- **Description.** Vous pouvez donner un court descriptif de votre liste ou catégorie de liens.

Notez qu'ici vous avez la possibilité de supprimer la catégorie fournie par défaut dans WordPress, à savoir la blogroll.

Sachez également que si vous supprimez une catégorie de liens, les liens qui lui sont associés ne seront pas supprimés mais attribués à la catégorie de liens par défaut, à savoir la blogroll.

Gestion et paramétrage des commentaires

Plus votre blog aura de succès, plus il recevra de commentaires. Il est donc important de savoir les gérer mais aussi de savoir les paramétrer. Les commentaires sont une des parties les plus intéressantes du blog car c'est grâce à eux que vont exister les discussions et les débats. Un blog sans commentaires ne serait pas vraiment un blog.

Malheureusement, ce sont également les commentaires qui vont être soumis aux attaques de spam, identique à celui que vous pouvez recevoir dans vos e-mails.

Ici, il va donc falloir distinguer et gérer les commentaires en provenance des visiteurs "humains", qui sont là pour engager une discussion, et des robots qui viennent polluer votre blog de spam.

Gestion des commentaires

Pour gérer vos commentaires au quotidien, rendez-vous sous l'onglet Commentaires. Vous retrouvez ici le même fonctionnement que pour les autres pages de gestion du blog. Voici une liste complète des spécificités de la gestion des commentaires :

- Vous pouvez afficher les commentaires selon leur état, qu'ils soient en attente de modération ou approuvés (voir Figure 3.66).



Figure 3.66
Affichage des commentaires selon leur état.

- Vous avez la possibilité, une fois le ou les commentaires sélectionnés, de leur donner un état "en masse", c'est-à-dire modifier plusieurs commentaires en même temps. Vous pourrez ainsi les approuver, les marquer comme indésirables, les désapprouver ou encore les supprimer, quel que soit leur nombre.

- WordPress fournit différentes informations sur la personne qui a laissé le commentaire : nom, site web, adresse e-mail et adresse IP (voir Figure 3.67).



Figure 3.67
Données concernant l'auteur du commentaire.

- Accédez à la page web de l'article sur laquelle celui-ci a été laissé en cliquant sur son lien.
- Accédez directement aux commentaires de cet article en cliquant sur le nombre de commentaires dans la colonne En réponse à cet article.
- Enfin, vous pouvez décider de l'action à faire pour ce commentaire : l'approuver/désapprouver, le rendre indésirable, le supprimer, le modifier, voire répondre directement au commentaire (voir Figure 3.68).



Figure 3.68
Actions sur les commentaires.

Notez que lorsqu'un commentaire apparaît en jaune, c'est qu'il est en attente de modération.

Paramétrage des commentaires

La gestion des commentaires peut paraître complexe si elle n'est pas bien paramétrée dès le départ. Bien entendu, nous parlons ici des blogs à forte fréquentation, mais même si vous avez un blog plus "intime", vous ne serez pas protégé du spam dans les commentaires.

WordPress fournit par défaut l'excellente extension Akismet, qui vous protégera la plupart du temps contre ce spam (voir au Chapitre 4 la section sur les extensions), mais la plateforme vous propose également toute une série de fonctionnalités qui vous faciliteront la vie au quotidien et vous permettront de dresser quelques barrières utiles contre le courrier non désiré.

Ces options se situent dans l'onglet Réglages et le sous-menu Discussion.

Réglages par défaut des articles

Figure 3.69
Réglages par défaut des articles.

Réglages par défaut des articles
- [ ] Tenter de notifier les blogs liés depuis cet article.
- [x] Autoriser les notifications depuis les autres blogs (notifications par pings et rétroliens)
- [x] Autoriser les visiteurs à publier des commentaires sur les articles

(Ces réglages peuvent être modifiés pour chaque article.)

Dans ce premier bloc, vous pouvez décider de notifier ou non les blogs que vous mentionnez dans vos articles (voir Figure 3.69). Vous avez également la possibilité d'accepter ou non les notifications en provenance d'autres blogs. Enfin, vous pouvez choisir d'autoriser ou non les commentaires pour l'ensemble du blog.

Autres réglages des commentaires

Figure 3.70
Autres réglages des commentaires.

Autres réglages des commentaires
- [x] L'auteur d'un commentaire doit renseigner son nom et son adresse de messagerie
- [ ] Un utilisateur doit être enregistré et connecté pour publier des commentaires
- [ ] Fermer automatiquement les commentaires pour les articles vieux de plus de 14 jours
- [x] Activer les commentaires imbriqués jusqu'à 5 niveaux
- [ ] Diviser les commentaires en pages, avec 50 commentaires de premier niveau par page et la dernière page affichée par défaut

Les commentaires doivent être affichés avec le plus ancien en premier

- L'auteur d'un commentaire doit renseigner son nom et son e-mail : si vous cochez cette case, le visiteur ne pourra pas laisser de commentaire s'il n'a pas renseigné au minimum un nom et un e-mail (voir Figure 3.70).
- Un utilisateur doit être enregistré et connecté pour publier des commentaires : ici, vous voulez que chaque participant à une discussion sur votre blog soit enregistré sur ce même blog pour pouvoir laisser un commentaire. C'est une solution qui risque de freiner les commentaires mais qui vous protégera complètement du spam.
- Fermer automatiquement les commentaires pour les articles vieux de plus de X jours : vous pouvez décider de fermer les commentaires pour tous les articles après une date fixée.
- Activer les commentaires imbriqués jusqu'à X niveaux : à partir de WordPress 2.7, il est possible d'avoir une gestion de commentaires imbriqués. Comme nous le verrons plus longuement au Chapitre 6, il est possible de répondre à un commentaire spécifique. Ici, vous pouvez donc décider du nombre de niveaux de discussion.
- Diviser les commentaires en pages, avec X commentaires par page et la X page affichée par défaut : ici, vous allez décider du nombre de commentaires à afficher par page.
- Les commentaires doivent être affichés avec le plus X en premier : à vous de choisir si vous souhaitez avoir le premier commentaire en haut de la page ou le dernier.

M'envoyer un e-mail lorsque

Figure 3.71
M'envoyer un e-mail lorsque.

M'envoyer un e-mail lorsque
- [x] Un nouveau commentaire est publié
- [x] Un commentaire est en attente de modération

WordPress vous propose ici de recevoir ou non un e-mail lorsqu'un commentaire a été laissé sur votre blog, ou lorsqu'il y en a un en attente de modération (voir Figure 3.71).

Il est très important de cocher cette case pour savoir ce qui se passe sur votre blog, sans pour autant avoir besoin de passer par l'interface d'administration.

Avant la publication d'un commentaire

Figure 3.72
Avant la publication d'un commentaire.

- Un administrateur doit toujours approuver le commentaire : aucun commentaire ne s'affichera sur le blog tant que l'administrateur ne l'aura pas validé manuellement (voir Figure 3.72). Pour un blog qui a peu de visites, cela peut être utile. Par contre, cela peut devenir rapidement ingérable si votre blog a beaucoup de succès.
- L'auteur d'un commentaire doit avoir déjà publié au moins un commentaire approuvé : cette notion est très intéressante. En effet, toute personne qui viendrait pour la première fois sur votre blog et qui laisserait un commentaire serait automatiquement mise en attente de modération. C'est un peu une manière de faire le tri et de voir le sérieux du commentateur. Par contre, si cette personne laisse un deuxième commentaire, celui-ci sera automatiquement approuvé et apparaîtra directement sur le blog.

Enfin, WordPress vous propose deux fonctionnalités qui vont mettre en place des conditions pour savoir quand un commentaire doit être mis en attente de modération, voire considéré directement comme indésirable.

Modération de commentaires

Figure 3.73
Modération des commentaires.

Tout d'abord, vous allez décider du nombre de liens maximal à partir duquel un commentaire doit être mis directement en attente de modération (voir Figure 3.73).

Nous savons que le spam ou pourriel utilise les blogs pour "vendre" des produits. Le spam fournit parfois toute une liste de liens. Fixer une limite basse peut être une solution pour minimiser son impact. Par défaut, ce nombre de liens est limité à 2.

File de modération

WordPress vous propose de créer une liste qui va contenir des mots, des noms, des URL, des adresses e-mail ou encore des adresses IP, pour lesquels les commentaires seront directement placés dans la file de modération. Ainsi quand un commentaire arrivera sur votre blog avec une ou plusieurs de ces indications, il sera automatiquement mis dans la file de modération.

Notez que cette fonction reconnaît également les bouts de mots et que vous ne devez mettre qu'une valeur par ligne.

Liste noire

C'est le même concept que la file de modération. Fournissez le même type de liste avec toujours les mêmes types d'informations, c'est-à-dire des mots, des noms, des URL, des adresses e-mail ou encore des adresses IP, mais cette fois-ci dans le but de les marquer définitivement comme indésirables. Les commentaires ne seront même pas mis en liste d'attente. Cela dit, vous avez toujours la possibilité de les approuver manuellement en allant les chercher dans l'onglet Commentaires indésirables, sous le menu de gestion des commentaires.

Ces options de liste de modération et de liste noire sont des options facultatives.

Avatars

Les avatars sont de petites vignettes qui s'affichent à côté de votre nom ou de votre commentaire. Le concept a été lancé il y a quelques années par Gravatar (**www.gravatar.com**). Vous créez un compte à votre nom et vous lui associez une image. À chaque fois que vous allez laisser un commentaire sur un blog, Gravatar va faire le rapprochement entre votre adresse e-mail et votre compte Gravatar et affichera votre image en conséquence (voir Figure 3.74).

En 2007, Automattic, la société qui édite WordPress, a racheté Gravatar. Et aujourd'hui, cette option est intégrée directement dans les options de discussion.

Vous disposez de plusieurs options pour gérer les avatars (voir Figure 3.75).

Figure 3.74
Affichage des avatars.



Figure 3.75
Gestion des avatars.

Affichage des avatars
- Ne pas afficher les avatars
- Afficher les avatars

Classement maximal
- G — Visibles par tous
- PG — Possiblement offensants, réservés normalement aux personnes de 13 ans et plus
- R — Réservés aux personnes de plus de 17 ans
- X — Réservés aux adultes

Avatar par défaut
Les utilisateurs n'ayant pas d'avatar peuvent se voir attribuer un logo générique, ou un avatar généré à partir de leur adresse e-mail.
- Homme mystère
- Blanc
- Logo Gravatar
- Identicon (Généré)
- Wavatar (Généré)
- MonsterID (Généré)

Tout d'abord, vous pouvez décider d'afficher ou non les avatars. Certains blogueurs apprécient les avatars, qui apportent plus de personnalisation aux commentaires, alors que d'autres pensent que c'est inutile et préfèrent garder le minimum.

Ensuite, les avatars sont classés par famille. Pour protéger les plus jeunes publics, vous pouvez décider d'afficher sur votre blog les avatars qui sont visibles par tous, et refusés ceux qui sont réservés aux personnes de plus de 17 ans.

Enfin, il arrive que certaines personnes qui viennent commenter vos articles ne possèdent pas de Gravatar. Dans ces cas-là, vous pouvez attribuer un avatar par défaut à ces visiteurs.

Différents rôles d'utilisateurs

Sous WordPress, il existe cinq différents types d'utilisateurs qui ont des rôles et donc des droits différents. Ces différents rôles sont :

- abonné ;
- contributeur ;
- auteur ;
- éditeur ;
- administrateur.

Voici un tableau récapitulatif des différentes actions qui leur sont attribuées.

Tableau 3.03 : Rôles des utilisateurs sous WordPress

Action	Abonné	Contributeur	Auteur	Éditeur	Administrateur
Écrire un article		X	X	X	X
Publier un article			X	X	X
Écrire une page				X	X
Gérer TOUS les articles				X	X
Gérer SES articles			X	X	X
Gérer les pages du blog				X	X
Gérer les catégories				X	X
Importer base de données					X
Exporter base de données					X
Gérer TOUS les commentaires				X	X
Gérer SES commentaires			X	X	X
Modérer les commentaires				X	X
Laisser des commentaires	X	X	X	X	X
Gérer la blogoliste				X	X
Gérer les thèmes					X
Gérer les plugins					X
Gérer les utilisateurs					X
Gérer les options					X

Créer un nouvel utilisateur

Le premier utilisateur WordPress est celui que vous avez créé lors de l'installation de votre blog. Il s'agit de l'utilisateur "admin" qui assume le rôle d'administrateur.

Il existe plusieurs manières de créer un nouvel utilisateur. Soit vous avez un compte d'administrateur, auquel cas vous avez tous les droits pour créer un utilisateur, soit vous permettez à n'importe qui de se créer un compte sur votre blog, à partir du moment où il connaît l'URL qui lui permettra de générer son profil et où vous l'aurez paramétrée de cette manière dans les réglages de WordPress.

... en tant qu'administrateur

Pour créer un utilisateur en tant qu'administrateur, vous devez vous rendre sous l'onglet Utilisateurs et cliquer sur le sous-menu Ajouter. Vous arrivez alors sur une nouvelle page qui va vous permettre de créer un nouvel utilisateur. Ici, il y aura trois informations obligatoires : l'identifiant, l'adresse e-mail et le mot de passe. Ensuite, vous avez la possibilité de fournir le nom, le prénom ou encore le site web de l'auteur. Enfin, vous allez devoir attribuer un rôle à chacun de vos nouveaux utilisateurs en fonction des droits que vous souhaitez leur conférer (voir Figure 3.76).

Figure 3.76
Créer un nouvel utilisateur.

... en tant que visiteur du blog

Comme nous l'avons fait remarquer précédemment, il est tout à fait possible d'autoriser n'importe quel visiteur à s'enregistrer. Cela est notamment très utile si vous souhaitez qu'un utilisateur soit connecté pour pouvoir laisser un commentaire.

Pour autoriser ces inscriptions, vous devez au préalable en faire le paramétrage dans l'onglet Réglages, sous le menu Général. Là, descendez jusqu'à la ligne Inscription (voir Figure 3.77).

Figure 3.77

Inscription et rôle du nouvel utilisateur.

Inscription	☐ Tout le monde peut s'enregistrer
Rôle par défaut de tout nouvel utilisateur	Abonné

En cochant cette case, vous autorisez n'importe quel visiteur à s'inscrire sur votre blog. Pour cela, vous devez lui présenter un lien d'inscription sur le blog. Ce lien est fourni par le thème default mais pas forcément par tous les thèmes. Le visiteur arrivera sur une nouvelle page où il pourra s'enregistrer comme nouvel utilisateur de votre blog.

Vous pouvez donc autoriser vos lecteurs à s'inscrire sur votre blog. Mais quel va être leur rôle ? C'est ce que vous allez paramétrer dans le champ suivant, Rôle par défaut de tout nouvel utilisateur.

Dans la majorité des cas, vous lui donnerez le rôle d'abonné. C'est d'ailleurs le rôle choisi par défaut par WordPress. Le nouvel utilisateur ne pourra alors laisser que des commentaires. Vous pouvez éventuellement lui donner un autre rôle par défaut, mais son rôle sera alors plus étendu. À vous de choisir en fonction de vos besoins.

Il est bon de noter ici que la plupart du temps les blogs n'utilisent pas cette fonction, laissant la possibilité aux visiteurs de faire un commentaire sans avoir besoin de s'inscrire.

Gestion des utilisateurs

En tant qu'utilisateur, vous avez le droit de modifier les informations vous concernant en allant sur l'onglet Utilisateurs et le sous-menu Votre profil. Ici, au-delà des informations données lors de la création de l'utilisateur, vous allez renseigner d'autres informations qui pourront être utilisées par le blog (voir Figure 3.78).

Tout d'abord, vous pouvez personnaliser l'interface d'administration de WordPress pour votre propre usage :

- Choisissez d'afficher ou non l'éditeur visuel pour la rédaction d'articles.
- Choisissez les couleurs de l'interface d'administration.

Ensuite, en plus de retrouver l'ensemble des informations vous concernant, vous allez fournir un pseudonyme et choisir le nom à afficher publiquement sur votre blog.

En dessous, vous pouvez fournir différentes informations pour vous contacter en dehors de l'adresse e-mail (ces options sont facultatives) : AIM, Yahoo Messenger et Jabber/Google Talk.

Figure 3.78
Votre profil d'utilisateur.

Ensuite, vous avez la possibilité de laisser quelques lignes personnelles sur vous. Cette information, une sorte de biographie, est de plus en plus utilisée sur les blogs multiauteurs. Son paramétrage est expliqué en détail dans la partie de cet ouvrage consacrée à la création d'un thème WordPress.

Enfin, vous pouvez modifier votre mot de passe utilisateur comme nous l'avons vu au début de ce chapitre.

Différents paramétrages du blog

Dans cette section, nous allons voir l'ensemble des paramétrages du blog qui n'ont pas encore été évoqués jusqu'ici mais que vous devez tout de même prendre en compte pour le bon fonctionnement de votre blog.

Tous les points examinés ici se paramètrent dans l'onglet Réglages.

Options générales

Titre et description du blog

Figure 3.79
Titre et description du blog.

Au moment de l'installation de WordPress, vous avez dû choisir un titre pour votre blog. Si vous souhaitez le modifier, c'est ici que vous le ferez (voir Figure 3.79).

Vous allez également en profiter pour donner un slogan, une description de votre blog, celle qui est proposée par défaut ne correspondant probablement pas à la thématique de votre site.

Adresse de WordPress et du blog

Figure 3.80
Adresse de WordPress et du blog.

WordPress vous donne la possibilité de positionner vos fichiers WordPress à une adresse différente de celle du blog (voir Figure 3.80). Si vous avez gardé le dossier wordpress tel que vous l'avez téléchargé, vous devriez avoir une URL de blog de cette forme : *www.exemple.fr/wordpress*.

Ici vous allez placer l'adresse de votre blog à un autre endroit de votre site, à sa racine, par exemple, c'est-à-dire sous la forme *www.exemple.fr*, tout en laissant les fichiers sous le dossier wordpress.

Comment procéder ?

1. Renseignez les nouvelles adresses pour vos fichiers WordPress ainsi que l'adresse à laquelle vous souhaitez voir apparaître le blog (1).
2. Avant d'aller plus loin, enregistrez les modifications.
3. Allez dans votre logiciel FTP sur votre hébergement et positionnez l'ensemble des fichiers de WordPress à l'endroit indiqué plus haut, dans les options générales.
4. Copiez les fichiers index.php et .htaccess se trouvant sous le dossier wordpress pour les placer à l'endroit où vous souhaitez voir apparaître le blog. Ici, nous voulons que le blog apparaisse à la racine du site (*www.exemple.fr*), nous allons donc placer ces deux fichiers à la racine de l'hébergement, en dehors du dossier wordpress.
5. Ouvrez le fichier index.php avec votre éditeur de texte favori et changez la ligne suivante :

   ```
   require('./wp-blog-header.php') ;
   ```

   en

   ```
   require('./wordpress/wp-blog-header.php').
   ```

6. Comme vous avez "remonté" le fichier index.php, il n'est plus au même niveau que le fichier wp-blog-header.php. Il faut donc modifier le chemin pour dire au moteur de blog d'aller chercher ce fichier dans le dossier wordpress.
7. Retournez maintenant sur l'interface d'administration de WordPress (*www.exemple.fr/wordpress/wp-admin/*) et allez sur l'onglet Permalien qui se trouve sous le menu Réglages. Là, étant donné que vous avez modifié l'URL des pages du blog, vous devez mettre à jour les permaliens pour que WordPress prenne bien en compte tous ces changements.

En effet, les URL des articles vont passer d'une structure telle que *www.exemple.fr/wordpress/identifiant-de-larticle* à *www.exemple.fr/identifiant-de-larticle*. Cliquez donc sur le bouton Enregistrer les modifications, tout en bas de la page, pour mettre ces permaliens à jour.

8. Pensez à faire cette modification très vite après l'installation du blog pour que les moteurs de recherche n'aient pas à réindexer tout votre site, ce qui pourrait être préjudiciable à votre référencement.

Adresse e-mail

Figure 3.81

Adresse e-mail à renseigner dans les réglages.

Adresse de messagerie
exemple@exemple.fr
Cette adresse n'est utilisée que pour l'administration du site ; par exemple, la notification de l'inscription d'un nouvel utilisateur.

L'adresse que vous fournissez ici n'est pas directement liée à votre compte utilisateur (voir Figure 3.81). C'est l'adresse e-mail utilisée pour l'administration du blog, celle que vous avez renseignée lors de la création du blog. Par exemple, lorsqu'un nouvel utilisateur est créé, c'est à cette adresse qu'une notification sera envoyée. Si par la suite vous voulez la modifier, c'est ici qu'il faudra le faire.

Dates et horaires

Figure 3.82

Formats dates et horaires.

Fuseau horaire
UTC+0
L'heure UTC actuelle est 2010-07-14 10:03:04
Choisissez une ville dans le même fuseau horaire que le vôtre.

Format de date
14 juillet 2010
2010/07/14
07/14/2010
14/07/2010
Personnalisé : j F Y 14 juillet 2010
Documentation sur le format des dates (en anglais). Cliquez sur « Enregistrez les modifications » pour mettre à jour l'exemple affiché.

Format d'heure
10 h 03 min
10:03
10:03
Personnalisé : G \h i ' 10 h 03 min

La semaine débute le
Lundi

Le temps est une donnée importante et incontournable sur un blog (voir Figure 3.82). Sans cette notion de chronologie, un blog ne serait plus un blog. Il est donc important de bien paramétrer le fuseau horaire sur lequel vous vous trouvez, ainsi que les formats de date et d'heure, pour permettre à vos lecteurs et visiteurs de savoir quand un article a été publié.

Le format de date proposé par défaut correspondra normalement à vos besoins. Si vous souhaitez le paramétrer différemment, d'autres formats vous sont proposés par défaut, et

vous avez même la possibilité de personnaliser cette date. Pour plus d'informations sur les formats de date, n'hésitez pas à aller sur cette page web : **http://ch.php.net/date**.

Vous pouvez également choisir le format pour l'heure, de la même manière que pour la date.

Enfin, on vous propose de choisir le jour qui débute la semaine. Dans nos contrées européennes, la semaine commence le lundi, mais dans certaines cultures, le premier jour de la semaine peut être différent.

Options d'écriture

Champ de saisie et mise en forme

Figure 3.83
Taille du champ de saisie et mise en forme.

Taille du champ de saisie 20 lignes
Mise en forme
☑ Convertir les émoticônes, comme :-) et :-P, en images lors de l'affichage
☐ WordPress doit automatiquement corriger les balises XHTML non valides

Dans WordPress, il est possible de choisir la taille du champ de saisie de vos articles (voir Figure 3.83). Ici, vous allez donc définir le nombre de lignes qui détermineront la hauteur du champ de saisie. Cette fonctionnalité peut se révéler utile, même si depuis la sortie de la version 2.5 de WordPress, vous avez la possibilité d'écrire les billets en format Plein écran (voir la section de ce chapitre consacrée à la rédaction de votre premier article pour plus de précisions).

Notez également que sur la page de rédaction d'un article, vous avez la possibilité d'étirer la fenêtre de saisie en "tirant" sur le coin en bas à droite (voir Figure 3.84).

Figure 3.84
Étirer la fenêtre d'édition d'un article ou d'une page.



Les options d'écriture proposent également quelques fonctionnalités concernant la mise en forme de vos articles :

- Possibilité de convertir les émoticônes en image. Les émoticônes sont ces petites têtes jaunes qui affichent des humeurs dans les billets. On les appelle aussi des "smilies" (ou *smileys*). Elles permettent de donner une intonation à certaines phrases. Vous en apprendrez plus sur cette page du Codex : **http://codex.wordpress.org/Using_Smilies**.
- Possibilité de laisser WordPress corriger automatiquement les balises XHTML qui seraient non valides. Cela s'adresse plutôt à ceux qui voudraient utiliser l'éditeur HTML et qui feraient des fautes lors de l'insertion des balises. Si vous cochez cette case, WordPress corrige automatiquement les erreurs de code que vous pourriez faire.

Catégories par défaut pour les articles et les liens

Figure 3.85
Catégories par défaut des articles et des liens.

Catégorie par défaut des articles Non classé
Catégorie par défaut des liens Liens

Vous définissez ici la catégorie par défaut utilisée lorsque vous oubliez d'en mentionner une au moment de la publication de vos articles (voir Figure 3.85). Le fonctionnement est identique pour déterminer une catégorie de liens par défaut.

Quand vous rédigez un article et que vous oubliez de mentionner une catégorie, celle qui est définie ici par défaut sera automatiquement prise en compte. C'est la même chose pour les liens. Si vous ne leur attribuez pas de "liste", celle qui est mentionnée ici sera utilisée par défaut.

Publier un article

Avec cette option de WordPress, vous pouvez avoir sur votre navigateur préféré un favori qui, lorsque vous cliquez dessus, vous permet d'arriver directement sur une page de rédaction d'un article (voir Figure 3.86). Pour cela, il vous suffit de cliquer sur le lien Publier un article et, tout en laissant le bouton de la souris enfoncé, de placer le lien dans la barre de votre navigateur.



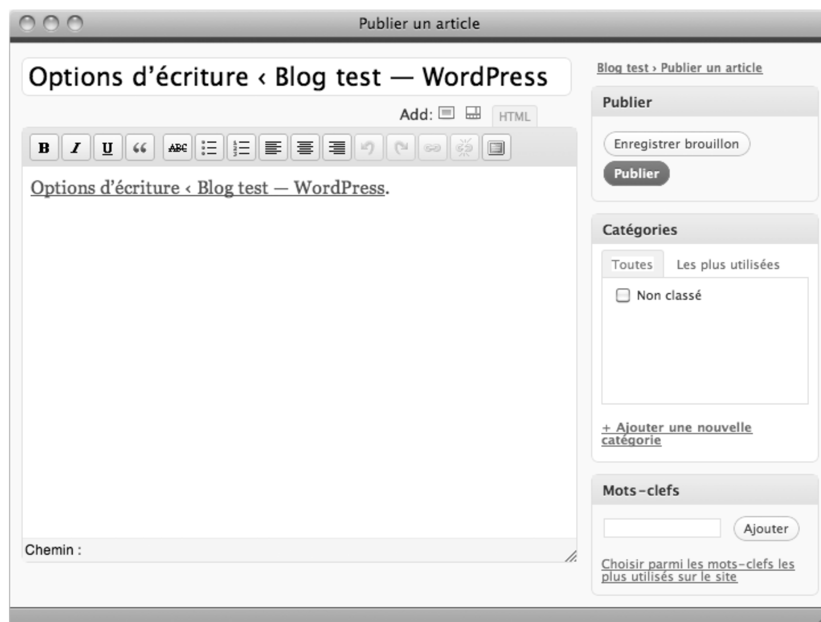
Figure 3.86
Favori dans la barre du navigateur.

Ainsi, dès que vous voudrez rédiger un article rapidement à partir d'une page web, vous cliquerez sur ce lien qui vous amènera directement sur une page spéciale de rédaction d'un article.

Prenons un exemple. Vous surfez sur le Web et vous trouvez un sujet d'article que vous souhaitez immédiatement bloguer. Vous restez sur cette page et vous cliquez sur le bouton Publier un article. Une nouvelle fenêtre va s'ouvrir sur une page dédiée de rédaction d'article, avec un titre préformaté avec le contenu de la page web sur laquelle vous êtes et avec un lien vers cette même page dans le contenu même de l'article (voir Figure 3.87).

Figure 3.87

Publier un article à partir d'une page web visitée.



Envoi d'article par e-mail

Figure 3.88

Envoi d'article par e-mail.

Envoi d'article par e-mail

Pour publier dans WordPress par e-mail, vous devez définir un compte e-mail secret avec une adresse POP3. Tout e-mail reçu à cette adresse sera publié. Il vaut donc mieux garder cette adresse à l'abri des regards. Voici trois chaînes aléatoires que vous pourriez utiliser : `Drs8uH8q`, `MsamgwDW`, `qTpUj4A2`.

Serveur de messagerie	mail.example.com Port 110
Identifiant	login@example.com
Mot de passe	password
Catégorie par défaut des articles envoyés par e-mail	Non classé

WordPress permet de publier des articles directement à partir d'un e-mail (voir Figure 3.88). Pour cela, vous devez renseigner les différentes informations de l'adresse e-mail utilisée pour publier vos articles :

- Serveur e-mail + port ;
- Identifiant ;
- Mot de passe ;
- Catégorie par défaut à attribuer aux billets arrivant par e-mail.

Ainsi, lorsque vous rédigerez des messages destinés à l'adresse e-mail paramétrée ci-dessus, ils seront automatiquement mis en ligne, dans la catégorie définie par défaut.

Publication à distance

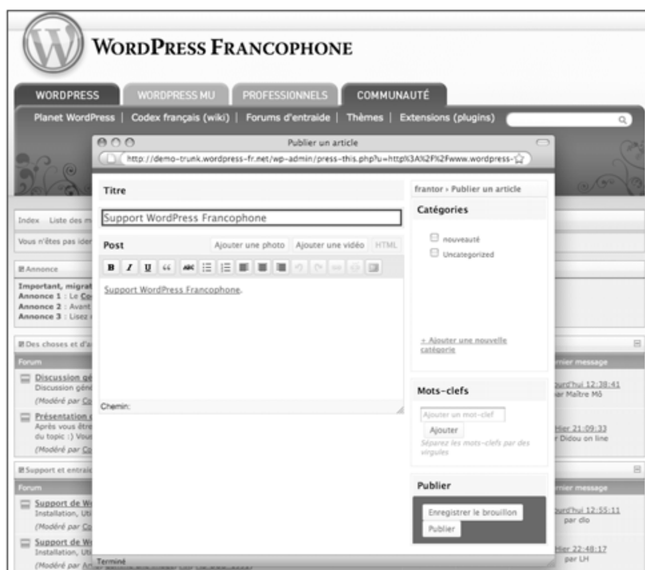


Figure 3.89
Publication à distance.

Il existe sur le marché un grand nombre d'outils qui permettent de gérer votre blog en dehors de l'interface d'administration, qui demande obligatoirement une connexion à Internet (voir Figure 3.89). Ces outils vous permettent donc de rédiger vos articles "offline", c'est-à-dire en local, sur votre ordinateur, sans être connecté. Nous pouvons citer notamment Livre Writer sur Windows et Ecto ou MarsEdit sur Mac OS X.

Cependant, pour publier des articles que vous avez rédigés sur une application tierce, vous devez passer par un protocole de publication Atom ou *via* une des interfaces de publication XML-RPC. Vous autorisez ainsi la communication entre les deux en choisissant le protocole qui correspond à vos besoins. De plus en plus d'éditeurs gèrent automatiquement le choix du bon protocole.

Services de mise à jour

Les blogs sont des sites web qui sont mis à jour régulièrement. Pas mal de services en ligne se sont créés autour de ce concept, et aujourd'hui des moteurs de recherche pour les blogs proposent notamment les derniers articles parus. C'est le cas de services tels que Technorati ou Google Blog Search qui sont beaucoup utilisés par les internautes.

Sur cette page, vous allez renseigner les adresses web de ces services pour les prévenir que vous avez publié un nouvel article. Cette notification se fera en même temps que la publication de l'article.

L'adresse à indiquer ici est celle qui va notifier le service. On l'appelle l'adresse de "ping". Par exemple, celle de Technorati est **http://rpc.technorati.com/rpc/ping** (vous trouverez

plus d'informations sur le Codex WordPress et notamment toute une liste mise à jour de services à notifier : **http://codex.wordpress.org/Update_Services**).

Par défaut, WordPress inclut l'adresse de pingomatic (**http://rpc.pingomatic.com/**), qui est un service qui va informer toute une série de moteurs de recherche que vous avez publié un nouvel article.

Attention toutefois à ne pas trop utiliser de services de ping car ils ont tendance à ralentir considérablement la publication des articles.

Options de lecture

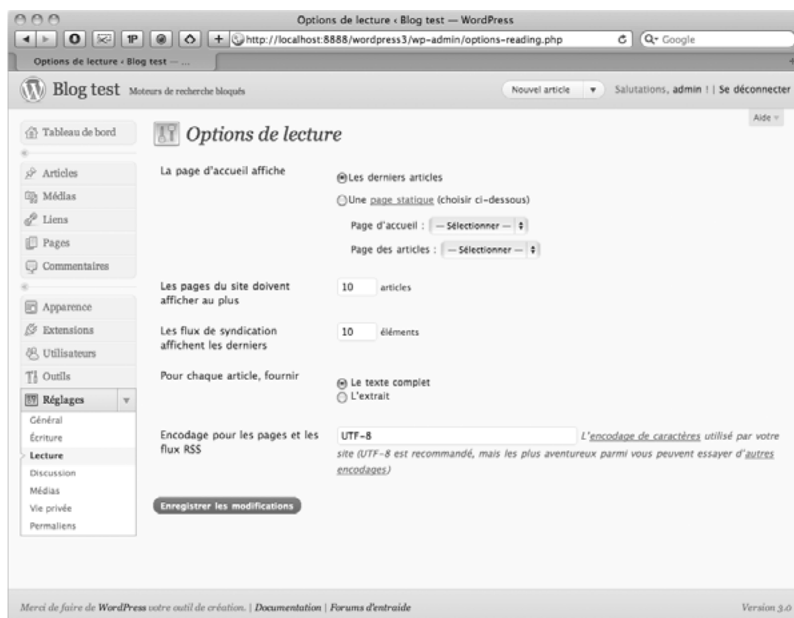


Figure 3.90
Options de lecture.

Les options de lecture apportent une touche de personnalisation à votre blog ainsi qu'à son flux RSS (voir Figure 3.90).

Quelle page d'accueil pour votre blog ?

Vous choisissez ici la page d'accueil de votre blog. Par défaut, cette page est celle qui affiche les derniers articles. Mais vous avez la possibilité de désigner une autre page de votre blog comme page d'accueil de votre site web. Cela peut être une page de présentation par exemple, et ainsi le blog devient une page parmi d'autres sur le site.

Comment procéder ? En fait, vous allez choisir quelle page sera votre page d'accueil et laquelle accueillera les articles.

Prenons un exemple. Admettons que vous souhaitiez présenter votre site sur votre page d'accueil et avoir une autre page dédiée à vos articles. Vous avez saisi vos différentes

informations sur la page À propos et c'est elle que vous allez choisir pour être la "page d'accueil" de votre blog.

Vous allez également créer une nouvelle page du blog, qui s'intitulera Articles et que vous choisirez comme page des articles (voir Figure 3.91).

Figure 3.91
Choix de la page d'accueil du blog.

La page d'accueil affiche
Vos derniers articles
Une page statique (choisir ci-dessous)
Page d'accueil : About
Page des articles : Articles

Validez vos choix en enregistrant vos modifications et allez voir le résultat directement sur votre blog. Vous avez maintenant une page d'accueil qui est une page statique, qui reprend le contenu de la page À propos, et vous avez une nouvelle page, intitulée Articles, qui vous donne accès à l'ensemble de vos articles.

Affichage du nombre d'articles sur les pages du blog

La page d'accueil de votre blog compte par défaut dix articles. Vous avez la possibilité d'en afficher plus ou moins selon vos envies et vos besoins.

Affichage des articles dans les flux de syndication

Si votre blog a du succès, il sera lu par un grand nombre de personnes *via* un lecteur de flux RSS tel que Netvibes (**www.netvibes.com**), Google Reader (**www.google.com/reader**) ou encore NetNewsWire (**www.newsgator.com/INDIVIDUALS/NETNEWSWIRE/**).

Ici, vous pouvez choisir le nombre d'articles à afficher dans votre flux. Par défaut, il affiche les dix derniers, mais vous pouvez modifier cette option. Décidez aussi si vous souhaitez proposer la totalité de vos articles ou seulement les premières lignes, incitant alors le lecteur à se rendre sur le blog pour lire la suite.

Concernant ces deux aspects, pas mal d'études et de sondages ont été réalisés sur Internet pour savoir ce que préféraient les lecteurs, ceux qui utilisent les lecteurs de flux RSS pour suivre différents blogs. La tendance pour le nombre d'articles à afficher se situe à une moyenne de dix à quinze articles. C'est surtout très utile pour le lecteur qui vient de s'abonner à votre blog et qui va avoir accès à ce nombre d'articles.

Par contre, les débats pour savoir s'il faut afficher l'article complet ou uniquement un extrait ont souvent été houleux, chacun défendant sa position. En fait, il y a d'un côté le blogueur qui voudrait bien que le lecteur vienne lire l'article directement sur le blog et de l'autre le lecteur qui, au contraire, aimerait bien consulter ses flux RSS sans avoir besoin de toujours aller lire la suite sur le blog.

Et c'est généralement le lecteur qui a le dernier mot. On a souvent vu des blogs passant en flux "tronqués" – ne proposant donc qu'un extrait de leurs articles – perdre un nombre

important de lecteurs car ces derniers jugeaient pénible de ne pas avoir accès directement à l'intégralité des articles *via* leur lecteur de flux RSS.

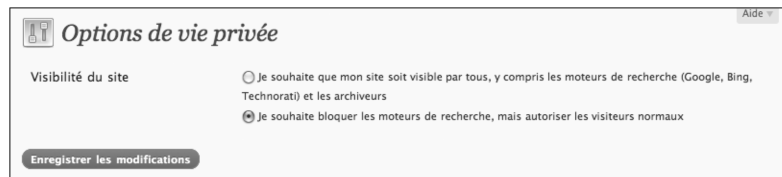
Enfin, un nombre grandissant de personnes utilisent leur téléphone portable pour consulter des flux RSS. Pour ces personnes, lire les articles directement dans le flux sera plus optimal que d'ouvrir le navigateur web du téléphone pour lire le reste de l'article.

Encodage des pages et des flux RSS

L'encodage par défaut sur WordPress est l'UTF-8. Vous avez la possibilité de le changer, mais ce n'est conseillé que si vous maîtrisez le développement web. Pour tout utilisateur lambda, cette fonctionnalité n'a pas lieu d'être modifiée.

Menu Vie privée

Figure 3.92
Menu Vie privée.



Au moment de l'installation de votre blog, on vous a demandé si vous souhaitiez apparaître dans les moteurs de recherche. La modification de votre présence sur Internet se fera *via* ce menu, qui se trouve sous le menu Réglages (voir Figure 3.92).

Modifier la structure des permaliens

Chacun de vos articles a une adresse web, une URL. Cette adresse est ce qu'on appelle un permalien. Et ce permalien peut prendre des formes différentes.

Par défaut, WordPress propose la structure de permalien suivante :

```
http://www.exemple.fr/?p=123
```

qui vous donne l'adresse du blog, puis l'identifiant de l'article en question, ici 123.

Le problème avec ce type de structure c'est que visuellement elle ne donne aucune information intéressante sur le contenu de votre blog.

WordPress propose quelques structures préétablies vous permettant de la modifier :

- Date et titre ;
- Mois et titre (la plus utilisée) ;
- Numérique.

Mais vous pouvez également définir vous-même votre structure sur la ligne Structure personnalisée.

Les mots-clefs et le référencement

Les moteurs de recherche utilisent beaucoup d'informations pour mieux référencer et indexer les blogs et les sites web en général. Parmi ces informations, il y a les mots-clefs. Ces mots, vous les retrouvez dans le contenu même de l'article ainsi que dans le titre, mais également dans l'URL.

Ainsi, si vous souhaitez avoir de nombreuses visites en provenance de Google par exemple, et avoir vos articles bien positionnés, il est important d'avoir une bonne structure de permaliens qui soit compréhensible et qui indique notamment le titre du contenu. WordPress permet donc de modifier cette structure en choisissant celle qui correspond le mieux à vos besoins.

Imaginons que vous souhaitiez voir apparaître deux informations dans votre permalien : la catégorie et le titre de l'article. Ces deux informations sont pertinentes et permettront à n'importe qui d'avoir un aperçu rapide du contenu de l'article. Vous allez donc utiliser ces deux marqueurs, `%category%` et `%postname%`, pour obtenir la structure suivante :

```
http://www.exemple.fr/%category%/%postname%/
```

L'URL prendra alors cette forme (exemple avec comme catégorie "webdesign") :

```
http://www.exemple.fr/webdesign/titre-de-mon-article
```

Une telle structure va permettre de récupérer les différents mots-clefs que sont webdesign et les mots composant le titre de l'article. Et ceci n'est qu'un exemple. WordPress vous permet de construire votre propre structure sur la ligne Structure personnalisée et en utilisant l'un des marqueurs suivants :

- `%year%` : correspond à l'année de publication de l'article sur quatre chiffres (par exemple 2008).
- `%monthnum%` : mois de l'année, sur deux chiffres (de 01 à 12).
- `%day%` : jour dans le mois, avec affichage de la date sur deux chiffres (de 01 à 31).
- `%hour%` : heure de la journée, sur deux chiffres (de 00 à 24).
- `%minute%` : minute dans l'heure, affichée sur deux chiffres (de 00 à 59).
- `%second%` : seconde dans la minute, affichée sur deux chiffres (de 00 à 59).
- `%postname%` : nous venons de le voir ci-dessus, il affichera les différents mots composant le titre de l'article, séparés par des tirets.
- `%post_id%` : affiche l'identifiant de l'article.
- `%category%` : affiche le nom de la catégorie de l'article (s'il y en a plusieurs, WordPress affiche la première par ordre alphabétique).
- `%author%` : affiche l'auteur de l'article.

Cas particuliers d'hébergeurs qui ne permettent pas la réécriture des permaliens

Dans certains cas, vous ne pourrez pas modifier la structure des permaliens comme cela est proposé ci-dessus, car votre hébergeur n'autorise pas le "mod_rewrite", c'est-à-dire qu'il ne permet pas dans le cas présent la réécriture des permaliens.

Si vous êtes dans ce cas précis, il y a une astuce qui vous permettra toutefois d'utiliser la structure de permaliens de votre choix.

Vous allez pouvoir utiliser les marqueurs voulus, mais vous devrez précéder la structure en y associant le fichier index.php qui permet l'affichage du blog en ligne. La structure aura alors la forme suivante :

```
http://www.exemple.fr/index.php/%category%/%postname%/
```

Ce n'est peut-être pas aussi esthétique que la structure "normale", mais cela vous permettra tout de même d'y afficher les informations souhaitées.

Préfixe des catégories et des tags

WordPress vous permet ici d'ajouter un préfixe pour les adresses web concernant vos catégories et vos tags. L'utilité principale est d'ajouter un mot-clef à votre URL. Cette option reste cependant optionnelle et peu utilisée.

L'onglet Outils

Publier un article

On retrouve ici la même option que celle décrite précédemment dans le sous-menu Écriture.

Convertisseur de catégories et de tags

WordPress a créé une extension qui vous permet de convertir vos catégories en tags ou l'inverse. En cliquant sur le lien, vous allez être amené sur la page Import du même onglet Outils. Là, vous cliquerez encore sur le lien du convertisseur, parmi les autres outils d'importation, et vous serez redirigé vers une page d'extension que vous pourrez télécharger. Ensuite, pour l'utiliser, vous ferez les mêmes manipulations que pour n'importe quelle extension.

Importer

Le menu Importation va vous permettre d'importer une base de données en provenance d'un autre blog WordPress ou en provenance d'autres plates-formes de blogging. Vous pourrez aussi récupérer de nombreuses tables utilisées par différentes extensions de WordPress.

Exporter

WordPress vous permet d'exporter vos articles, sous un format WordPress eXtende RSS ou WXR. Vous pouvez également filtrer ce que vous voulez exporter. Ainsi, vous pouvez choisir la date de début à prendre en compte et/ou la date de fin. Vous pouvez faire un tri par auteur, catégories, types de contenu ou encore par statuts d'articles.

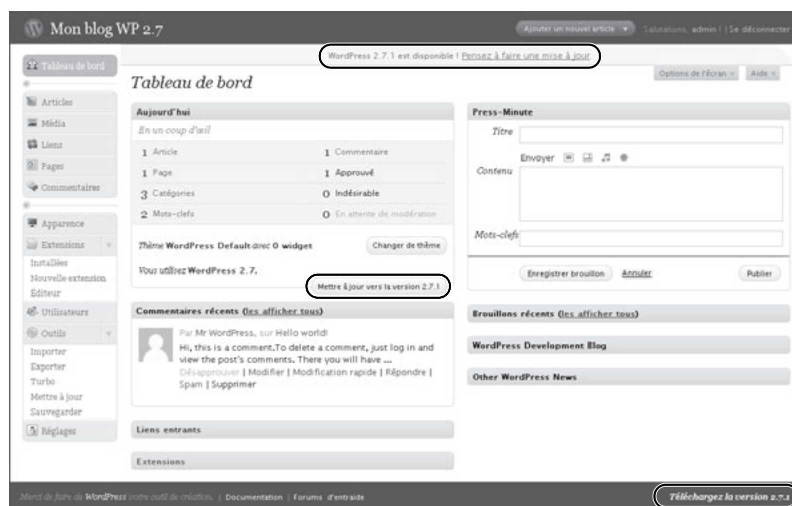
Mettre à jour WordPress

WordPress n'est pas un logiciel qui se repose sur ses lauriers. Bien qu'il soit devenu l'un des outils de gestion de blog les plus utilisés au monde, ses développeurs n'ont de cesse de chercher à améliorer la vie du blogueur par l'ajout de nouvelles fonctionnalités et l'amélioration de celles déjà en place, grâce aux versions majeures et mineures du logiciel.

Par ailleurs, comme tout logiciel, WordPress n'est pas parfait, et ces versions majeures sont souvent suivies de versions mineures qui corrigent certains défauts ou s'assurent que la sécurité est optimale. La disponibilité d'une version supérieure à la vôtre est clairement indiquée dans l'interface d'administration (voir Figure 3.93).

Figure 3.93

Le bandeau d'information notifiant l'existence d'une nouvelle version.



Dans les faits, il sort en moyenne trois versions majeures par an, et chacune est suivie d'au moins une version mineure corrigeant les menus bogues résiduels. Cela suppose donc pour le blogueur consciencieux environ six mises à jour par an. Même si les versions mineures n'impliquent le plus souvent que quelques fichiers, il est vrai que procéder à toutes ces mises à jour peut facilement devenir laborieux, voire contraignant.

Heureusement pour nous (et pour vous), la mise à jour de WordPress est tout aussi facile et rapide que son installation, pour peu que vous suiviez correctement les instructions. Nous vous présentons ici les méthodes principales : la mise à jour automatique, la mise à jour manuelle qui se fait *via* un client FTP et qui est la méthode adoptée par la grande majorité des utilisateurs, la mise à jour *via* Subversion, qui est à réserver aux utilisateurs avertis et qui savent ce qu'ils font, et la mise à jour *via* une extension comme WordPress Automatic Upgrade.

Seuls les utilisateurs d'une version de WordPress égale ou supérieure à la 2.7 pourront profiter de la mise à jour automatique, les utilisateurs de versions plus anciennes (jusqu'à la

version 2.6.3) ainsi que les utilisateurs de WordPress 2.7 dont l'hébergement n'autorise pas la mise à jour automatique devront se reposer sur les autres méthodes.

Mise à jour automatique

L'un des principaux apports de la version 2.7 de WordPress est l'apparition d'un outil de mise à jour automatisé directement intégré à l'interface d'administration. Celui-ci prend totalement en charge le transfert des fichiers depuis le serveur de développement vers l'hébergement du blog, rendant obsolète le processus classique de téléchargement, décompression et mise en ligne *via* le client FTP.



L'outil de mise à jour automatique, apparu avec WordPress 2.7, n'est évidemment utilisable que pour mettre à jour vers des versions qui lui sont supérieures : de 2.7 à 2.7.1, de 2.7.1 à 2.8, de 2.8 à 2.9, etc. Il n'est évidemment pas disponible pour mettre à jour vers la 2.7. Par conséquent, les blogueurs qui souhaiteraient en profiter devront faire une dernière mise à jour selon leur méthode habituelle.

Le processus de mise à jour automatique est conçu de telle sorte qu'une mise à jour qui aura échoué ne sera pas prise en compte :

1. Téléchargement de l'archive depuis WordPress.org dans le dossier local /wp-content.
2. Décompression de l'archive localement, dans le dossier /wp-content/upgrade/ /core/wordpress.
3. Création du fichier temporaire .maintenance à la racine du blog. Tant que ce fichier existe, le blog ne peut recevoir de nouvelle visite, tout comme son interface d'administration, ce qui permet de ne pas troubler le fonctionnement de WordPress tandis que les fichiers sont mis à jour (voir Figure 3.94).
4. Copie des nouveaux fichiers en remplacement des anciens.
5. Mise à jour de la base de données.
6. Effacement du contenu du dossier /wp-content/upgrade.
7. Effacement du fichier .maintenance.
8. Effacement des fichiers obsolètes.



Le thème par défaut est également mis à jour ; par conséquent, si votre blog utilise le thème par défaut avec des modifications personnelles, assurez-vous que vous avez bien renommé son dossier dans `/wp-content/themes`, sinon vous perdrez ces modifications. Par ailleurs, l'outil de mise à jour est conçu pour effacer tous les anciens fichiers et dossiers qui ne sont plus pris en compte par WordPress ; faites donc une sauvegarde de votre dossier `/wp-images` s'il se trouve que vous vous en servez encore…

Figure 3.94
Le message de maintenance, bloquant l'accès au blog.

Briefly unavailable for scheduled maintenance. Check back in a minute.

Lancer la mise à jour

Il y a deux moyens d'ouvrir l'écran de mise à jour automatique : en choisissant l'option Mettre à jour du sous-menu Outils, ou en cliquant directement sur le lien Veillez à rester à jour, qui se trouve en bas de chaque page de l'interface d'administration de WordPress.

L'écran qui s'affiche alors vous avertit tout d'abord qu'il est toujours utile de faire une mise à jour des fichiers de votre blog et de la base de données. Certaines extensions, comme WP-DB-Backup, peuvent vous y aider.

Le lancement de la mise à jour se fait en cliquant sur le bouton Mettre à jour automatiquement. À partir de là, deux possibilités peuvent se présenter, qui dépendent de votre hébergement :

- WordPress vous demande vos identifiants FTP (voir Figure 3.95).
- La mise à jour se déclenche directement.



Techniquement, WordPress fait appel à l'outil suPHP pour permettre la mise à jour en direct sans problème de droits d'accès. Si cet outil n'est pas installé sur le serveur, ou si ses droits ont été révoqués par l'utilisateur, plusieurs outils sont utilisés en fonction de la configuration serveur pour assurer une mise à jour avec un maximum de sécurité : ftpext, ftpsocket ou ssh2.

Figure 3.95
Les identifiants FTP dépendent de votre hébergeur.

Demo WordPress - Developpement Version

Informations sur connexion

Pour lancer la requête demandée, vous devez donnez les informations requises.

Nom de l'Hôte :

Identifiant

Mot de passe

Type de connexion

FTP

FTPS (SSL)

Continuer

Si votre hébergeur vous demande vos identifiants FTP, saisissez ceux que vous avez utilisés pour installer WordPress. Dès que vous les avez saisis, le processus se lance et il est identique dans les deux cas (voir Figure 3.96).



Figure 3.96
Les messages affichés pendant une mise à jour complète.

Une fois la mise à jour achevée avec succès, vous pouvez tranquillement retourner dans les différentes sections de WordPress pour bloguer comme d'habitude…

Mise en garde

Le processus de mise à jour automatique vous décharge totalement de la laborieuse étape de remplacement des fichiers : c'est le serveur PHP qui s'occupe de tout. Mais il est toujours possible que des problèmes côté serveur bloquent la mise à jour chez certains hébergeurs. En effet, les huit étapes du processus demandent beaucoup de mémoire (pour décompresser l'archive .zip, entre autres choses), un temps maximal d'exécution PHP approprié, et en général une configuration PHP adéquate.

De fait, la mise à jour pourra tout simplement être impossible chez certains hébergeurs : mémoire insuffisante, pare-feu trop restrictif, droits d'accès aux fichiers limités, nécessité de changer de configuration, etc. Souvent, vous ne pouvez le savoir qu'en essayant…

Par exemple, par défaut, un hébergement 1&1 (**1and1.fr**) ne pourra lancer la mise à jour que si le site utilise la configuration PHP 5, la configuration PHP 4 étant insuffisante. Pour changer de configuration, il suffit d'ajouter la ligne suivante au fichier .htaccess à la racine du compte : `AddType x-mapp-php5 .php`.

Chez un autre hébergeur populaire, OVH (ovh.fr), la mise à jour automatique n'arrive également pas à son terme (particulièrement avec la configuration PHP 5, activée en plaçant la ligne `SetEnv PHP_VER 5` dans le fichier .htaccess). Les clients d'OVH devront donc *a priori* en rester aux méthodes classiques…

La communauté WordPress a créé une page sur le Codex pour lister les résultats de mise à jour automatique selon les différents hébergeurs : ceux chez qui la mise à jour marche sans problème, ceux chez qui elle ne marche que dans certains cas, et ceux chez qui elle ne marche malheureusement pas. Vous pouvez y accéder *via* cette adresse : **http://codex.wordpress.org/Core_Update_Host_Compatibility**. C'est un wiki, donc n'hésitez pas à vous inscrire pour y contribuer par vos propres résultats de test !

Les possibilités d'échec sont donc réelles et, selon les cas, il est possible que vous ne puissiez faire appel à la mise à jour automatique. Vous devrez alors vous reposer sur la mise à jour manuelle ou *via* Subversion, ou à l'aide d'une extension comme WordPress Automatic Upgrade.

Mise à jour manuelle

WordPress s'est forgé une réputation (méritée) de facilité d'installation : les développeurs promettent aux utilisateurs un blog en état de marche en moins de 5 minutes. Dans le même ordre d'idée, la mise à jour de WordPress se fait idéalement en trois étapes, avec un minimum de règles à suivre.

Toutes les étapes sont à suivre impérativement pour s'assurer une mise à jour sans souci.

Étape 1 : préparation de la mise à jour

1. **Faites une sauvegarde.** Pour prévenir tout problème pendant la mise à jour, il est chaudement recommandé de télécharger l'ensemble des fichiers de WordPress présents sur votre compte FTP et de les mettre en sécurité sur votre disque dur. Si la mise à jour venait à échouer, vous pourriez ainsi revenir à la version précédente en effaçant tous les fichiers et en les remplaçant par cette sauvegarde.

   Pareillement, une exportation des données stockées dans MySQL peut éviter de mauvaises surprises. Pour cela, des extensions dédiées existent, comme WP-DB-Backup, **http://wordpress.org/extend/plugins/wp-db-backup/** (voir Figure 3.97). Pour ceux qui savent s'en servir, phpMyAdmin permet également d'exporter le contenu de la base dans de nombreux formats…

   N'oubliez pas cette étape : si la mise à jour casse votre blog et si vous perdez vos données, vous ne pourrez vous en prendre qu'à vous-même…

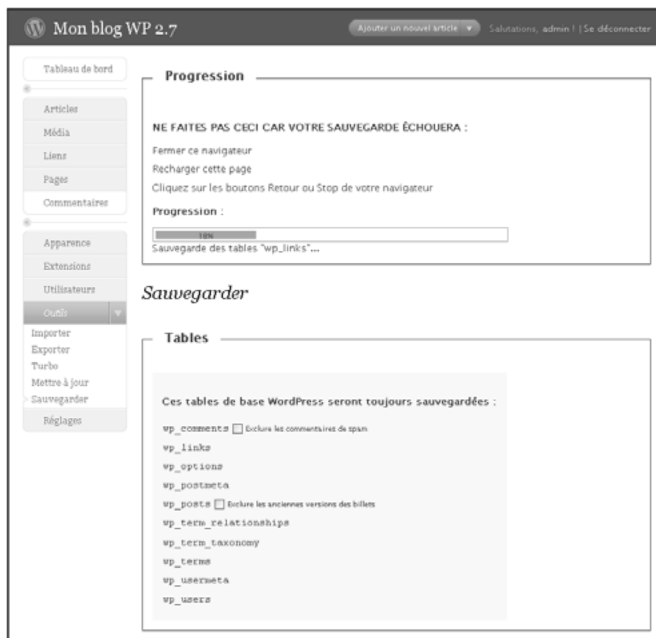


Figure 3.97
WP-DB-Backup en action.

2. **Désactivez toutes les extensions.** Certaines extensions peuvent être liées à WordPress au point de créer un conflit pendant la mise à jour. Les désactiver toutes permet de parer à tout problème. Pour ceux qui ont un grand nombre d'extensions, il suffit de cocher la case en tête du tableau des extensions pour toutes les sélectionner, puis de cliquer sur Désactiver. Après la mise à jour, il est conseillé de réactiver les extensions une à une pour s'assurer qu'aucune ne crée de conflit avec la nouvelle version. Lorsque c'est fait, si l'une des extensions se révèle défectueuse au point de vous empêcher de vous reconnecter à l'interface d'administration, la solution est de renommer son fichier sur le serveur, le temps de vous connecter et de la désactiver…

Étape 2 : remplacer les fichiers de WordPress

1. **Récupérez la dernière version.** Téléchargez la dernière version de WordPress en français depuis http://fr.wordpress.org, et décompressez l'archive sur votre disque dur.
2. **Nettoyez le serveur FTP.** Pour éviter les problèmes de fichiers non remplacés, connectez-vous sur votre compte FTP et effacez totalement les dossiers /wp-admin et /wp-includes. Ne touchez pas au dossier /wp-content (et /wp-images s'il existe), ni aux dossiers que vous auriez vous-même créés. Effacez également tous les fichiers à la racine du dossier de WordPress (index.php, wp-rss.php, etc.), SAUF les fichiers suivants (voir Figure 3.98) :
   – wp-config.php, qui contient les codes de connexion avec la base MySQL ;
   – .htaccess, qui contient les règles de réécriture pour obtenir des URL lisibles ;
   – les fichiers que vous avez installés vous-même, et ceux créés par des extensions, par exemple les scripts de statistiques ou un fichier sitemap.xml.

Effacez enfin certains sous-dossiers de /wp-content :

– /wp-content/cache ;

– /wp-content/plugins/widgets.

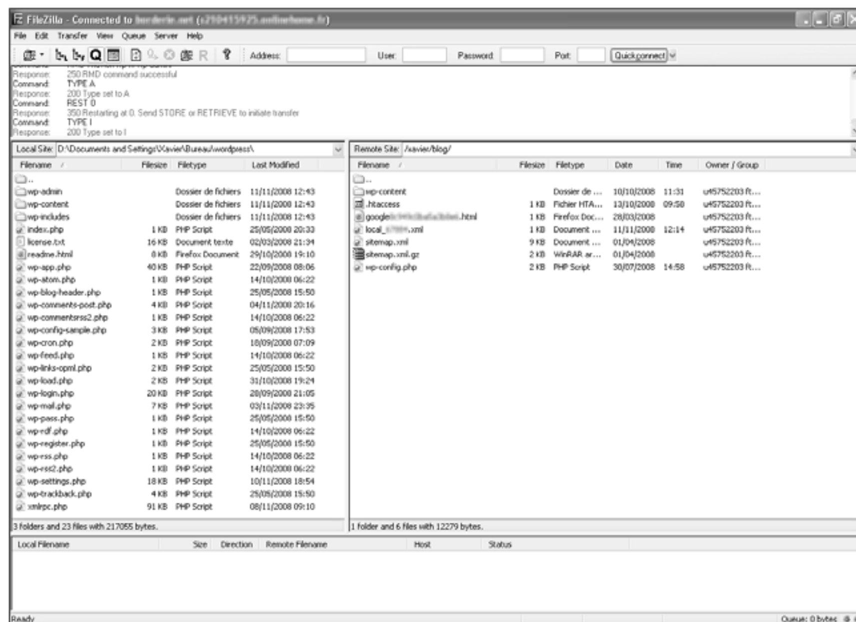


Figure 3.98

Votre hébergement après l'effacement des fichiers et dossiers remplaçables.

3. **Mettez en ligne les nouveaux fichiers.** Au moyen de votre client FTP, transférez les dossiers /wp-admin et /wp-includes ainsi que tous les fichiers à la racine depuis votre disque dur vers votre compte FTP.

   Ensuite, ouvrez sur votre compte FTP le dossier /wp-content et faites les modifications suivantes :

   – Remplacez l'extension Akismet par la version de l'archive.

   – Effacez et remplacez le thème par défaut Kubrick, dans /wp-content/themes/default, sauf si vous avez modifié ce thème vous-même, auquel cas n'y touchez surtout pas au risque de perdre ces modifications.

   – Mettez à jour le fichier de traduction française fr_FR.mo dans /wp-content/languages/.

   – Ne touchez pas aux autres extensions et thèmes sans savoir ce que vous faites.

Étape 3 : lancer et vérifier la mise à jour

Les nouveaux fichiers de WordPress sont maintenant en place sur votre compte FTP, mais il est possible que la nouvelle version nécessite également des modifications sur la base de données pour fonctionner correctement. Ces modifications sont réalisées par un script interne, upgrade.php, dont le lien vous sera proposé lorsque vous vous reconnecterez à

l'interface d'administration de WordPress après une mise à jour. Cliquez sur ce lien pour terminer le processus de mise à jour (voir Figure 3.99). Si le lien ne vous est pas proposé, vous pouvez le lancer en allant directement à l'adresse **http://votre-domaine.com/votre-dossier/wp-admin/upgrade.php**.



Figure 3.99
Message affiché par upgrade.php.

Si ces étapes ont été correctement suivies, votre version de WordPress est maintenant à jour. Il vous reste à parcourir les pages de votre blog et de l'interface d'administration pour vous assurer que tout fonctionne correctement – et découvrir les nouveautés de cette version !

Parmi les points à vérifier, vous pouvez lister :

- **Les permaliens des articles et pages.** Si votre blog est configuré pour avoir des permaliens lisibles, assurez-vous qu'ils fonctionnent toujours en parcourant quelques pages de votre blog.
- **La compatibilité des extensions.** Si vous avez bien suivi l'étape 1, toutes vos extensions ont été désactivées lors de la mise à jour. Une fois le script upgrade.php correctement lancé, il est temps de réactiver vos extensions, mais pas toutes à la fois. En effet, il est possible qu'une extension ancienne ne soit pas compatible avec la nouvelle version de WordPress et fasse réagir votre blog ou l'interface d'administration de manière inattendue.

  Pour découvrir l'extension fautive, réactivez donc les extensions une à une, et vérifiez qu'elles fonctionnent comme prévu. Le cas échéant, partez en quête d'une alternative à l'extension fautive, et contactez son auteur.
- **La compatibilité de votre thème.** Tout comme les extensions, un thème peut faire appel à des fonctions de WordPress depuis longtemps abandonnées. Assurez-vous donc que votre thème s'affiche comme avant. Le cas échéant, contactez son auteur, corrigez-le vous-même, ou cherchez une alternative plus récente.
- **La variable `SECRET_KEY`.** Si vous mettez à jour depuis une version inférieure à la 2.5, vous devez ajouter une ligne à votre fichier wp-config.php, afin de renforcer la sécurité de votre code : `define('SECRET_KEY', 'a-remplacer-par-une-suite-de-caracteres-au-hasard');` (voir Figure 3.100).

  Vous pouvez générer une clé secrète au hasard grâce au script hébergé sur le site de WordPress : **http://api.wordpress.org/secret-key/1.0/**.

Figure 3.100

Mettre en place la clé secrète dans wp-config.

```
/**#@+
 * Clefs uniques d'authentification.
 *
 * Modifiez ces valeurs pour des phrases uniques !
 * Vous pouvez les générer avec {@link http://api.wordpress.org/secret-key/1.1/ le service de clef
secrète de WordPress.org}
 *
 * @since 2.6.0
 */
define('AUTH_KEY', 'L&3@.ofxHW$\"[r]>0kl/z&z>(`z|Ypw$Z;F?/%&@:&JE~(|@*iPbds6RQTP.Ne%u');
define('SECURE_AUTH_KEY', 'hUH,V:x;$QDO+%ucH\"c[EUTM}UO$G2xNR\"-W2<6dD.#+.c+lI)tSBex*VA/&}/:{');
define('LOGGED_IN_KEY', 'H\'KRzK{~j@9d$(,D~C[CsG%l2<o=-w--7v=\\1d1* v9vy.OS{N\"nfZ~pd}:TCV{J');

/**#@-*/
```

Votre blog est maintenant à jour. Félicitations !

Mise à jour par Subversion

La mise à jour *via* FTP est suffisamment rapide pour pouvoir être faite en 5 minutes, mais devoir réitérer ces manipulations pour chaque nouvelle version peut rebuter certains utilisateurs, surtout ceux qui gèrent de nombreux blogs. Il est possible de réaliser automatiquement les mises à jour sans disposer de la version 2.7 de WordPress, et ce avec un minimum d'intervention, si vous utilisez directement le serveur Subversion de WordPress géré par Automattic, **http://svn.automattic.com/wordpress/**.

Cette méthode est à réserver aux utilisateurs qui connaissent Subversion et/ou les commandes, et qui ont la main sur leur serveur. Les fichiers passent directement des serveurs de WordPress à votre propre serveur, sans devoir faire une transition par votre disque dur, ni gérer une archive .zip (voir Figure 3.101).

Figure 3.101

Subversion en action dans la ligne de commande.

```
Terminal — bash
A    wp-admin/css/theme-editor-rtl.css
A    wp-admin/css/dashboard-rtl.css
A    wp-admin/css/theme-editor.css
A    wp-admin/css/dashboard.css
A    wp-admin/page-new.php
A    wp-admin/setup-config.php
A    wp-admin/link-manager.php
A    wp-admin/install.php
A    wp-admin/widgets.php
A    wp-admin/sidebar.php
A    wp-admin/link-parse-opml.php
A    wp-admin/import.php
A    wp-admin/options-permalink.php
A    wp-admin/page.php
A    wp-admin/options-writing.php
A    wp-admin/export.php
A    wp-admin/plugin-editor.php
A    wp-admin/edit-pages.php
A    wp-admin/admin-footer.php
A    wp-admin/categories.php
A    wp-admin/themes.php
A    wp-feed.php

Fetching external item into 'wp-content/plugins/akismet'
A    wp-content/plugins/akismet/akismet.gif
A    wp-content/plugins/akismet/akismet.php
A    wp-content/plugins/akismet/readme.txt
Checked out external at revision 73313.

Checked out revision 9611.
```

Subversion (SVN) est un outil de gestion de version utilisé par les développeurs de WordPress, qui repose sur un serveur librement accessible par le Web depuis un client dédié, mais

surtout *via* une connexion SSH que nous utiliserons ici. Ce serveur donne accès à toutes les versions existantes de WordPress correctement étiquetées (**http://svn.automattic.com/wordpress/tags/**), aux développements en cours pour une version donnée (**http://svn.automattic.com/wordpress/branches/**) [voir Figure 3.102], ainsi qu'à la dernière version en cours de développement dans le "tronc" (**http://svn.automattic.com/wordpress/trunk/**).

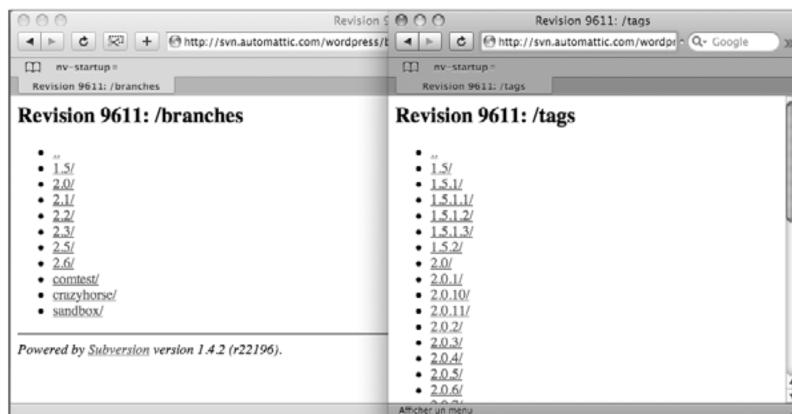


Figure 3.102
À gauche, le contenu de /branche ; à droite, celui de /tags.

Le tronc est logiquement instable, il peut changer du jour au lendemain, et n'est à utiliser que pour les utilisateurs qui ont un intérêt à être à la pointe du développement. Dans la vaste majorité des cas, il sera préférable de mettre à jour vers une version stable.

Les versions stables sont développées dans les branches dédiées (par exemple **http://svn.automattic.com/wordpress/branches/2.6/** pour la série 2.6.x) et sont souvent aussi peu stables que /trunk. Les véritables versions stables, une fois validées à partir du code dans /branches, sont placées dans /tags avec le numéro complet de la version (par exemple **http://svn.automattic.com/wordpress/tags/2.6.1/** pour la version 2.6.1, tirée des développements de la série 2.6.x).

Avant toute manipulation, vous devez savoir quelle version utiliser. Pour cela, visitez le dossier /tags avec votre navigateur et notez le numéro de la dernière version.

Installer un nouveau blog avec Subversion

Un blog ne peut être mis à jour avec Subversion que s'il est déjà géré par cet outil. Le mieux pour y arriver est donc de commencer dès le début par créer le blog en récupérant les fichiers de WordPress à l'aide de la commande de "checkout" de Subversion (`svn co`), ce après vous être placé dans le dossier de destination.

La manipulation est à faire depuis une interface de ligne de commande, une fois connecté au serveur web. Voici comment installer WordPress 2.6 par ce biais (voir Figure 3.103) :

```
$ mkdir blog
$ cd blog
$ svn co http://svn.automattic.com/wordpress/tags/2.6.1 .
```

Figure 3.103

svn co en action.

```
Terminal — svn
ordinateur-de-xavier-borderie:Sites xavier$ mkdir blog
ordinateur-de-xavier-borderie:Sites xavier$ cd blog
ordinateur-de-xavier-borderie:blog xavier$ svn co http://svn.automattic.com/wordpress/tags/2.6.3 .
A    wp-pass.php
A    wp-rss.php
A    wp-comments-post.php
A    wp-login.php
A    wp-blog-header.php
A    wp-rdf.php
A    wp-load.php
A    wp-includes
A    wp-includes/class-snoopy.php
A    wp-includes/default-filters.php
A    wp-includes/feed-rss2.php
A    wp-includes/post-template.php
A    wp-includes/plugin.php
A    wp-includes/locale.php
A    wp-includes/feed-atom.php
A    wp-includes/wlwmanifest.xml
A    wp-includes/category.php
A    wp-includes/taxonomy.php
A    wp-includes/template-loader.php
A    wp-includes/update.php
A    wp-includes/functions.wp-scripts.php
A    wp-includes/compat.php
A    wp-includes/class-pop3.php
A    wp-includes/class-phpmailer.php
```

N'oubliez pas le point final de la dernière ligne, séparé de l'URL, sinon la commande `svn` créera un dossier /tags/2.6.1 plutôt que placer tous les fichiers téléchargés à la racine du dossier créé avec la commande `mkdir`.

Une fois le transfert de fichiers terminé, renommez wp-config-sample.php en wp-config.php, et modifiez-le pour y mettre la configuration de votre serveur MySQL. Enfin, suivez le processus d'installation normal tel qu'il est décrit au Chapitre 2.

Mettre à jour le blog installé avec Subversion

Les fichiers de votre blog peuvent à présent être gérés par Subversion. À partir de maintenant, le processus de mise à jour est considérablement simplifié, éliminant le téléchargement de l'archive sur votre disque dur, l'envoi de fichiers sur le serveur FTP, etc. Tout cela grâce à la commande "switch" de Subversion (`svn sw`), qui modifie uniquement les fichiers différents.

Par exemple, pour mettre à jour vers WordPress 2.6.3 :

```
$ cd blog
$ svn sw  http://svn.automattic.com/wordpress/tags/2.6.3 .
```

Ici encore, n'oubliez pas le point à la fin de la ligne. Subversion devrait ainsi ne pas toucher à vos fichiers (extensions, thèmes, etc.), mais que cela ne vous empêche pas de faire une sauvegarde avant, au cas où le processus échouerait.

Une fois le transfert de fichiers achevé, suivez le processus normal de mise à jour expliqué dans ce chapitre, à commencer par une connexion à votre interface d'administration pour lancer le script upgrade.php.

Convertir un blog installé à la main en blog géré avec Subversion

Avec Subversion, la rapidité de la mise à jour est telle que nombreux sont ceux qui voudront s'en servir, même s'ils n'ont pas installé leur blog *via* Subversion dès le départ. Heureusement, il est très facile de réparer cet oubli.

Suivez ce processus pas à pas :

1. Faites une installation avec Subversion dans un dossier temporaire, par exemple blog-temp, comme nous l'avons vu précédemment.
2. Placez vos fichiers personnels (wp-config.php, .htaccess, les thèmes, extensions, images, scripts, fichiers de traduction, etc.) depuis votre dossier existant vers le dossier temporaire.
3. Pour conserver le dossier existant en cas de problème, renommez-le plutôt que de l'effacer, par exemple en "blog-vieux". Puis, renommez le dossier temporaire avec le nom normal du dossier du blog.
4. Lancez le script upgrade.php. Pour plus de sécurité, vous pouvez faire une sauvegarde de votre base MySQL.

Vous obtenez ainsi, à partir d'un blog installé à la main, les mêmes bénéfices qu'un blog installé avec Subversion.

Si un problème survient, il vous suffit de redonner au dossier "blog-vieux" son nom normal et de remettre en place vos données MySQL.

Mettre à jour au moyen d'une extension

L'extension WordPress Automatic Upgrade (**http://wordpress.org/extend/plugins/wordpress-automatic-upgrade/**) a servi d'inspiration à l'outil interne de mise à jour automatique (voir Figure 3.104). Elle a donc l'avantage de fonctionner pour les versions de WordPress inférieures à la 2.7.

Cette extension tente de simplifier le processus de mise à jour en recréant l'idée d'un script serveur qui télécharge et installe directement les fichiers de la nouvelle version, tout en étant plus simple à prendre en main. Il est sans doute plus sûr, étant donné que sa première action est de vous proposer de télécharger tous les fichiers de WordPress et le contenu de la base de données, avant de lancer la suite du processus (qui comprend une désactivation automatique de toutes vos extensions).

Pour autant, elle ne doit pas vous faire oublier de réaliser une sauvegarde de vos fichiers et de votre base de données avant de vous lancer dans la mise à jour.

WPAU se charge ensuite de mettre à jour les fichiers essentiels, à savoir le contenu des dossiers /wp-admin et /wp-includes, et les fichiers à la racine.



Figure 3.104
Administration de WordPress Automatic Upgrade.

En cas de problème

WordPress est depuis longtemps conçu pour que ses mises à jour se fassent rapidement et avec un minimum de souci pour l'utilisateur, quel que soit son niveau. Pourtant, la grande variété des configurations serveur et logicielle fait qu'une mise à jour peut, dans de rares cas, faire plus de mal que de bien. Voici quelques solutions aux problèmes les plus courants…

Le blog est bloqué en état de maintenance

La mise à jour automatique ne s'est pas achevée *a priori*, et un fichier .maintenance doit toujours se trouver à la racine de WordPress. Il est probable que la configuration par défaut de votre hébergement soit insuffisante pour la mise à jour automatique. Essayez de modifier cette configuration avec les possibilités offertes par votre hébergeur, ou contactez-le directement pour savoir s'il est possible qu'il modifie sa configuration PHP pour autoriser le déroulement complet de la mise à jour.

Pour accéder à nouveau à votre blog et à son interface d'administration, effacez simplement le fichier .maintenance au moyen d'un client FTP. Pour aller jusqu'au bout, effacez les archives .zip téléchargées dans le dossier /wp-content, et effacez le contenu du dossier /wp-content/upgrade.

N'hésitez pas à faire profiter la communauté de votre expérience en ajoutant votre hébergeur sur le wiki de WordPress : **http://codex.wordpress.org/Core_Update_Host_Compatibility**.

J'ai perdu les modifications de mon thème/d'une extension

Heureusement, vous avez pensé à faire une sauvegarde de vos fichiers avant de lancer la mise à jour, comme c'était conseillé ! Si ce n'est pas le cas, vous ne pouvez malheureusement que vous en prendre à vous-même…

La mise à jour automatique efface et remplace de nombreux fichiers et dossiers obsolètes. Parmi eux se trouvent :

- **Les thèmes Classic et Default.** Le contenu des dossiers /wp-content/themes/classic/ et /wp-content/themes/default/ est intégralement mis à jour. Donc, si vous avez personnalisé l'un des deux thèmes par défaut, par exemple en le traduisant ou en modifiant l'image d'en-tête, assurez-vous d'avoir déplacé vos fichiers modifiés dans leur propre dossier, par exemple /wp-content/themes/default2/.
- **Les extensions Akismet, Hello Dolly, Markdown et Textile.** Akismet (/akismet/akismet.php) est régulièrement mise à jour, donc son extension WordPress profite également d'un remplacement. L'extension Hello Dolly (hello.php) est moins souvent mise à jour, mais si vous vous en servez, son fichier sera aussi remplacé. Quant aux extensions Markdown (markdown.php) et Textile (textile1.php), elles datent également des premières versions de WordPress et sont aujourd'hui considérées comme obsolètes, donc effacées. À vous de faire en sorte de les conserver si vous vous en servez…

La mise à jour automatique a pour but de nettoyer vos fichiers WordPress et elle peut se montrer trop efficace au goût de certains. N'oubliez donc jamais de faire une sauvegarde de vos fichiers originaux.

Le blog ne s'affiche pas comme avant

Après une mise à jour, il peut arriver que le thème du blog apparaisse "cassé" : lignes de code PHP qui apparaissent, informations manquantes, etc. Le plus souvent, le code fautif provient d'une extension trop ancienne qui utilise du code qui n'est plus pris en compte par WordPress. Retrouvez-la en désactivant toutes les extensions, puis en les réactivant une à une. Une fois l'extension fautive trouvée, désactivez-la et cherchez une alternative compatible – et, idéalement, contactez l'auteur de l'extension défectueuse pour lui signaler le problème.

C'est plus rare, mais il peut arriver que la cause d'un thème "cassé" soit le thème lui-même, si celui-ci est trop ancien et donc basé sur du code obsolète. Dans ce cas, vous pouvez soit mettre la main dans le cambouis si vous avez des connaissances en HTML, soit contacter l'auteur du thème pour lui demander poliment de corriger le problème. Au pire, il vous reste toujours la possibilité de donner un nouveau look à votre blog, en utilisant l'un des nombreux thèmes disponibles sur le web. (Vous en trouverez une sélection sur **http://apprendre-wordpress.fr/livre.**)

J'ai réécrit certaines parties de WordPress

Certains utilisateurs préfèrent modifier directement le code de WordPress plutôt que d'écrire une simple extension. C'est une mauvaise idée, car les mises à jour vous obligent à garder la trace de vos modifications, et ne pas mettre à jour vous expose à des failles de sécurité pouvant faire des ravages. Plongez-vous dans la partie de ce livre qui explique le développement d'extensions, afin de transposer votre hack en une extension propre, dont vous pourrez même faire profiter la communauté…

Je préfère la version précédente

Bien que cela soit fortement déconseillé pour des raisons de sécurité, il est possible de revenir à une version précédente – dans la limite des différences entre deux versions.

En effet, si le retour en arrière entre versions mineures (de 2.6.1 à 2.6 par exemple) ne pose pas de problème, il n'en est pas de même entre les versions majeures, tant les modifications internes peuvent être importantes. Par exemple, une fois passé à une version supérieure à la 2.3, il devient délicat de revenir en arrière… et dangereux.

Pour revenir à une version antérieure, il suffit d'exploiter la sauvegarde des fichiers et de la base de données que vous avez faite lors de l'étape 1 de la mise à jour (automatique ou manuelle). Effacez tous les fichiers de WordPress et remplacez-les par les fichiers sauvegardés, puis effacez toutes les tables MySQL liées à WordPress et restaurez votre sauvegarde de base de données (à l'aide de l'outil phpMyAdmin par exemple).

Sans ces sauvegardes, revenir à une version précédente est presque impossible.

Je veux mettre à jour depuis une très vieille version

Le script upgrade.php prend en compte de nombreuses anciennes versions de WordPress, mais si votre blog utilise une version vraiment ancienne, comme la 1.2, le processus de mise à jour est sensiblement différent, du fait de l'introduction du système de thèmes dans la version 1.5. Il vous faudra donc télécharger WordPress 1.5, faire une mise à jour vers cette version, adapter votre thème, puis faire une nouvelle mise à jour depuis la version 1.5 vers la toute dernière.

Notez que les extensions et thèmes que vous utilisez devront être certainement mis à jour eux aussi, voire remplacés par des alternatives plus modernes…

4 Choisir le thème et les extensions pour son blog

Jusqu'à présent, nous avons insisté sur l'importance que peut avoir le paramétrage du blog. Dans ce chapitre, nous allons nous intéresser à l'apparence de votre blog. Nous passerons en revue les données qui sont à prendre en compte pour bien choisir un thème, c'est-à-dire un design pour votre blog. Nous verrons aussi comment personnaliser votre colonne latérale en utilisant les widgets, et enfin nous aborderons la notion d'extension et nous verrons lesquelles sont indispensables pour votre blog.

Bien choisir le thème pour son blog

La recherche d'un thème est sûrement l'un des aspects les plus intéressants lorsqu'on crée son propre blog. Mais cela peut également être l'un des plus difficiles. Il faut trouver le thème qui s'accordera bien avec le contenu et avec vos goûts. Et parfois, cela se révèle être une mission plutôt périlleuse. Nous allons ici vous donner quelques pistes qui vous permettront de trouver le style qui vous correspondra le plus et qui reflétera le mieux l'identité de votre blog.

Quel style pour mon blog ?

Lorsqu'on commence à parler de design pour un blog, on distingue deux catégories de personnes. Il y a tout d'abord celles qui savent ce qu'elles veulent et qui ont une idée précise de ce qu'elles recherchent. Et puis il y a celles qui ne savent pas du tout quel design choisir pour leur blog et qui retireront un grand bénéfice des pages qui vont suivre.

La communauté WordPress est gigantesque et elle regorge d'idées. Vous pourrez aussi bien parcourir différents sites qui proposent des thèmes WordPress que des sites qui référencent les plus grands blogs WordPress. Toutes ces ressources vous donneront un premier aperçu de ce que vous pouvez obtenir et vous guideront dans vos choix. Plus loin dans ce chapitre, nous vous donnerons une liste complète de ces ressources.

Mais, avant de partir à l'aventure, il est important que vous considériez quelques questions qui orienteront votre choix du meilleur thème pour votre blog. En effet, même si vous avez une idée du résultat que vous voudriez obtenir, vous devez toujours prendre en considération le visiteur et la corrélation entre le contenu et le contenant d'un blog. Voici quelques questions types que vous pourriez vous poser :

- Quel est le contenu de mon blog ? Est-ce un blog personnel ? Un blog professionnel ? Un photoblog ? Ou encore un blog d'actualités ?
- Quel design choisir en fonction de ce contenu ? Il existe de nombreux styles qui correspondent à différentes orientations. Ces derniers temps, la mode semble être aux thèmes

dits "magazines", qui correspondent parfaitement à des blogs de "news", d'actualités, mais qui vont très mal s'associer avec un contenu personnel ou un blog professionnel.

- Quelles couleurs choisir, toujours en fonction de ce contenu ? Des couleurs plutôt chaudes ? Plutôt froides ?
- Combien de colonnes pour mon blog ? Une colonne qui donne un aspect plus intime, plus personnel ? Deux colonnes qui sont un grand classique du blog ? Trois colonnes, voire plus, qui vont s'adresser plus à des blogs d'actualités ou professionnels ?
- Quelle place réserver au contenu par rapport à la ou les barres latérales ?

Si ces questions ne sont que des exemples parmi d'autres, elles vous permettront néanmoins d'orienter votre réflexion. N'hésitez pas non plus à parcourir le Web pour dénicher des idées. Trouvez le design qui correspond le mieux à vos envies, votre personnalité mais aussi au contenu de votre blog.

Enfin, pour tenter de trouver le thème parfait s'il existe, n'hésitez pas à télécharger tous ceux qui vous plaisent et essayez-les sur votre blog. C'est le meilleur moyen de voir celui qui vous conviendra le mieux.

Gratuit ou payant ?

Dans l'univers des thèmes WordPress, on trouve de tout. Il y a du bon comme du moins bon, et il y a du gratuit comme du payant. Le tout est de savoir ce que vous recherchez et si vous êtes prêt à débourser un peu d'argent pour vous payer un thème un peu plus évolué.

En fait, tout dépend du besoin. Parmi les thèmes gratuits, vous trouverez tout type de style. Par contre, du côté des thèmes payants (appelés également "thèmes premium"), vous retrouvez bien souvent des thèmes prévus pour des sites d'actualités ou des sites institutionnels. Cela paraît logique. Dès lors que vous développez un thème pour un blog et que vous souhaitez le rendre payant, autant le destiner à ceux qui auront les moyens de le payer et surtout qui verront cet achat comme un investissement.

Mais, si vous souhaitez créer un blog personnel prévu uniquement pour votre entourage, il est fort probable que vous trouviez votre bonheur sans avoir besoin de débourser quoi que ce soit. Par contre, si vous souhaitez créer un véritable site Internet, plutôt professionnel, et qui utiliserait des techniques avancées de WordPress, n'hésitez pas à vous tourner vers des solutions payantes.

Mais, là encore, faites bien attention. De nombreux développeurs et designers se sont lancés dans le design de thèmes WordPress, mais pas toujours de manière bien sérieuse. On trouve ici encore pas mal de thèmes qui sont vendus très cher et qui ne valent pas forcément le prix payé.

Dans la section suivante, vous trouverez une liste complète de sites web où vous pourrez vous procurer des thèmes gratuits ou payants de très bonne qualité. Certains sont vraiment des références en la matière et quel que soit le thème que vous téléchargerez, vous serez assuré de sa qualité.

À propos des "theme frameworks"

Tout d'abord, qu'est-ce qu'un "theme framework" : WordPress permet depuis quelques versions de créer des thèmes enfants à partir d'un thème principal. Ce thème principal est souvent construit sous forme de "framework". Il a une structure et des fonctionnalités qui vont permettre à l'utilisateur de s'en servir de manière poussée et optimale. Bien souvent, ces thèmes ne sont pas utilisés directement tels quels mais le sont à travers un thème enfant qui reprend la structure de son thème parent.

Quels en sont les avantages ?

Le thème parent, le "theme framework", ne va pas être modifié. Il pourra donc facilement être mis à jour. C'est un gros avantage pour bénéficier des nouvelles fonctionnalités du thème ou alors des évolutions de WordPress, le tout sans avoir à aller modifier le code du thème directement.

Quels sont les inconvénients ?

Le souci avec les thèmes frameworks est que ce sont de véritables usines à gaz. Donc, si vous souhaitez une fonctionnalité particulière qui n'existe pas dans le thème, il vous faudra mettre les mains dans le code, et là, vous risquez de vous heurter à des structures très complexes, difficiles à comprendre et à maîtriser pour quelqu'un qui ne connaît pas le PHP.

Ressources web

Répertoire de thèmes chez WordPress.org (http://wordpress.org/extend/themes/)

Pendant longtemps, WordPress a possédé un site web dédié aux thèmes : **http://themes.wordpress.net/**. Puis, pour des raisons diverses, le site n'a plus été maintenu.

Mais, début 2008, l'équipe WordPress a décidé de remettre le navire à flot avec un nouveau site, annexe du site principal de WordPress : **http://wordpress.org/extend/themes/**. Disons que c'est le site "officiel" pour les thèmes WordPress. La qualité peut sembler assez aléatoire, mais le système de navigation est très bien conçu et vous permettra de trouver facilement ce que vous cherchez.

À noter que vous avez la possibilité d'installer les thèmes présents sur le site officiel directement à partir de votre interface d'administration. Nous verrons comment faire dans une section suivante.

Woothemes (http://www.woothemes.com)

Woothemes propose à la fois des thèmes payants et gratuits. C'est aujourd'hui le numéro 1 en terme de ventes et de communauté. Le service existe depuis 2008 et propose régulièrement des thèmes de très bonne qualité avec certains conçus par des web designers célèbres.

Elegant Themes (http://www.elegantthemes.com)

Ici, tous les thèmes sont payants mais l'auteur propose un service qui est très intéressant. En effet, vous payez un peu moins de 20 dollars par an et vous avez accès à tous les thèmes qu'il propose. Et, quand on voit la qualité de ses thèmes, nul doute que cela vaut le coup. Vraiment un coup de cœur graphique !

Blog Themes Plus (http://blogthemesplus.com)

Ce site référence à la fois des thèmes payants et des thèmes gratuits. Vous y trouverez le meilleur des deux mondes. À noter que le site propose également des sites pour d'autres plates-formes telles que Tumblr ou Blogger.

Press75 (http://press75.com)

Ce site a été créé par Jason Schuller, un designer WordPress célèbre. Il propose à la fois des thèmes gratuits et payants de très bonne qualité. Les thèmes Press75 sont très appréciés pour leur univers graphique.

Studiopress (http://www.studiopress.com/)

Brian Gardner est l'auteur des thèmes proposés sur ce site. Brian est un des premiers concepteurs de thèmes WordPress. Il est considéré comme un pionnier et a proposé divers services au fur et à mesure de l'évolution de WordPress. Aujourd'hui, il est tourné vers les "themes frameworks" dont on a parlé précédemment. Il propose donc un thème de base et des thèmes enfants, le tout étant payant.

Thesis (http://diythemes.com/thesis/)

Ce site propose un thème qui pourrait s'apparenter à un framework. Il est d'ailleurs interprété différemment chez différentes personnes. C'est un thème simple graphiquement mais très poussé au niveau des fonctions. Si vous recherchez un thème qui vous donne toute liberté au niveau des options, Thesis pourrait être le bon choix.

Smashing Magazine (http://www.smashingmagazine.com/)

Smashing Magazine n'est pas à proprement parler un site de thèmes WordPress mais en propose régulièrement sur son blog. Les thèmes proposés sont toujours gratuits et de très bonne qualité. Le site propose également régulièrement des articles recensant les meilleurs thèmes WordPress.

Avec ces différentes ressources, vous devriez facilement trouver votre bonheur. Si ce n'était pas le cas, n'hésitez pas à utiliser les moteurs de recherche et à naviguer sur le Web. Internet regorge de thèmes pour WordPress. Mais, comme nous l'avons dit précédemment, tous ne sont pas toujours bien codés, certains sont sponsorisés et comprennent parfois des liens vers des sites de spam. En piochant dans la liste proposée ci-dessus, vous avez la garantie que votre blog aura un thème de qualité, tant au niveau du design qu'au niveau de son développement.

Installer et gérer son thème

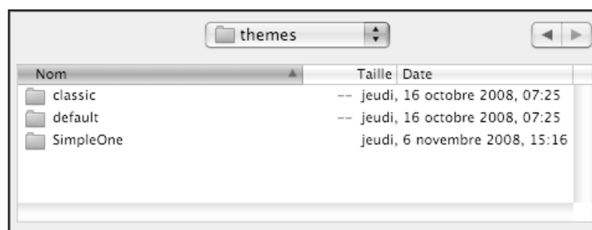
Installation d'un thème

À partir de l'hébergement

L'installation d'un thème pour WordPress est très simple. Suivez les instructions suivantes et vous profiterez rapidement de votre blog avec son nouveau design :

1. Une fois que vous avez trouvé le thème que vous voulez essayer, téléchargez-le sur votre ordinateur. Il est composé d'un dossier qui regroupe l'ensemble des fichiers du thème.
2. Ouvrez votre logiciel FTP préféré et connectez-vous à votre hébergement, au niveau du dossier des thèmes : wp-content > themes.
3. Transférez le thème dans ce dossier. Les fichiers se trouvent maintenant dans wp-content/themes/nomdutheme/ (voir Figure 4.01).

Figure 4.01
Dossier thèmes.



4. Une fois le transfert effectué, allez sur l'interface d'administration de WordPress, sous l'onglet Apparence et le sous-menu Thèmes. La page est scindée en deux parties : Thème courant, qui correspond au thème employé actuellement sur le blog, et Thèmes disponibles qui sont les thèmes que vous pouvez utiliser. C'est là que vous allez voir apparaître le thème que vous venez de transférer.
5. Descendez au niveau des Thèmes disponibles et cliquez sur la vignette du nouveau thème (voir Figure 4.02). Vous pouvez avoir un aperçu de votre blog avec le thème choisi avant de valider ce choix. Si vous êtes satisfait du résultat, cliquez en haut à droite de la fenêtre sur Activer, suivi du nom du thème (voir Figure 4.03).

La page se rafraîchit automatiquement et votre thème apparaît désormais comme thème courant (voir Figure 4.04).

Figure 4.02

Choisir le thème à afficher.



Figure 4.03

Activer le thème choisi.

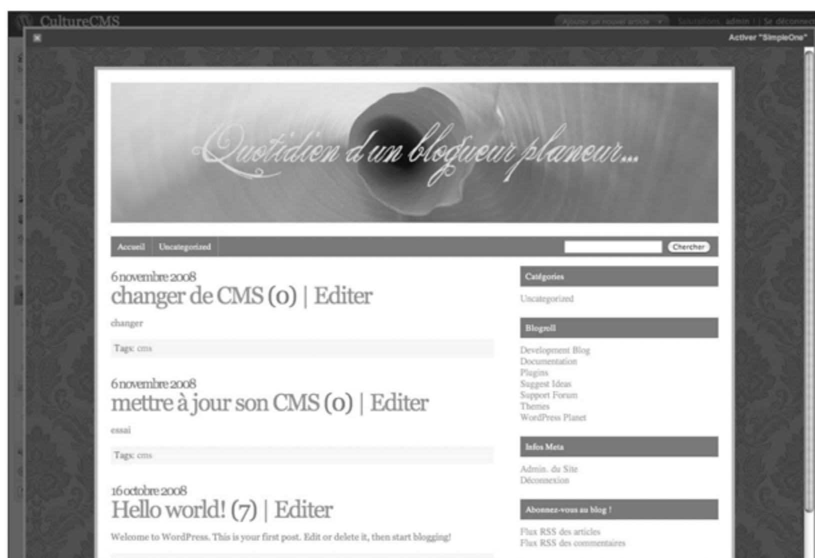


Figure 4.04

Thème courant.

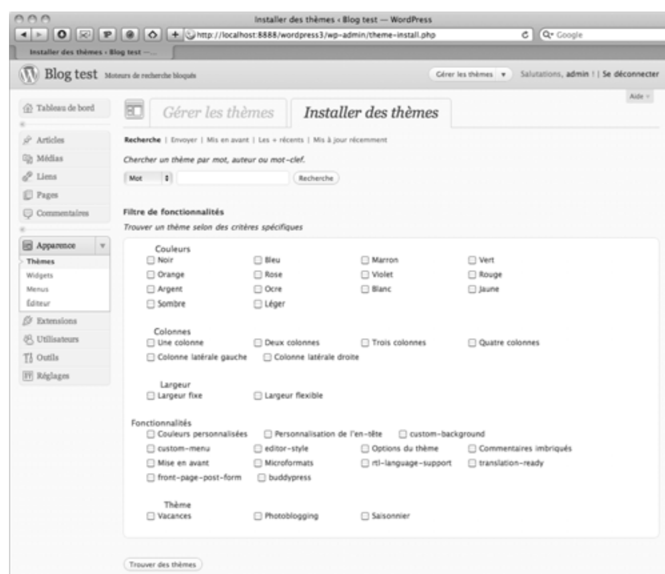


À partir de l'interface d'administration

Lorsque vous êtes dans le menu Apparence de l'administration de votre blog, vous avez deux onglets : Gérer les thèmes et Installer des thèmes. Cliquez sur ce dernier onglet pour voir apparaître une nouvelle page qui va vous permettre de chercher des thèmes en fonction de vos besoins (voir Figure 4.05).

Figure 4.05

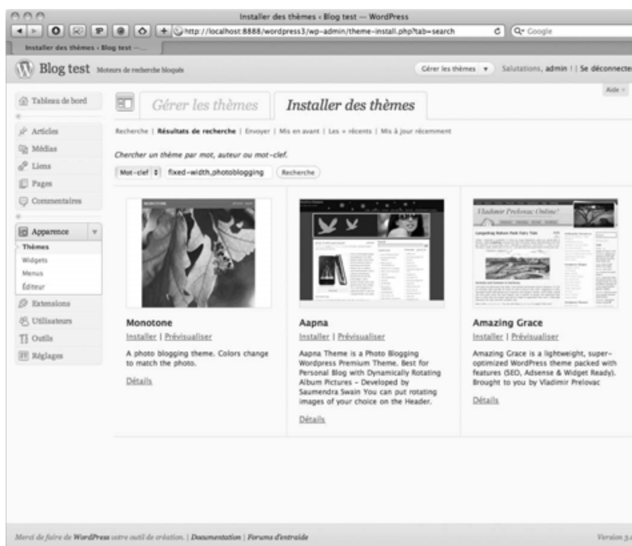
Installer des thèmes.



Une fois vos choix effectués, cliquez sur le bouton Trouver des thèmes. Là s'ouvre une nouvelle page avec une sélection de thèmes correspondant, ou presque, à vos attentes (voir Figure 4.06).

Figure 4.06

Différents thèmes disponibles en fonction des choix effectués.



Ce qui est intéressant à partir de cette page, c'est que vous allez pouvoir prévisualiser chacun des thèmes proposés sans avoir besoin de les télécharger. Ce n'est qu'une fois que vous aurez trouvé votre bonheur que vous cliquerez sur le bouton Installer, situé sous le titre du thème.

À partir de là, WordPress va installer le thème. Cela peut prendre jusqu'à quelques minutes, selon le thème et votre connexion Internet. Une fois le thème installé, vous pourrez une nouvelle fois le visualiser (mais cette fois-ci avec vos données) ou alors l'installer directement.

Modification d'un thème

Si vous connaissez bien le XHTML, CSS et autre PHP, vous aurez probablement envie de faire quelques modifications dans le thème que vous utilisez. Ceux-ci étant tous ou presque en licence libre, vous avez tout à fait le droit de les personnaliser comme bon vous semble. Veillez toutefois à ne pas supprimer les crédits pour l'auteur.

Si vous souhaitez effectuer des modifications sur votre thème, vous avez deux possibilités :

- Utiliser un éditeur de texte pour modifier les différents fichiers du thème.
- Utiliser l'éditeur de thème *via* l'interface d'administration de votre blog (Apparence > Éditeur de thème). Cette page vous permet de modifier chacun des fichiers composant votre thème, mais aussi les autres thèmes disponibles sur votre blog.

Pour plus d'informations concernant la structure d'un thème WordPress, reportez-vous au Chapitre 5 de cet ouvrage, portant sur le développement d'un thème pour WordPress.

Utilisation des widgets

Définition

Un widget est un petit module qui vient se placer dans la barre latérale de votre blog. Sur beaucoup de thèmes, vous avez déjà plusieurs blocs dans la barre latérale. Cependant, ce sont des morceaux de code insérés directement dans les fichiers PHP du thème. Les widgets, eux, s'installent directement à partir de l'interface d'administration de WordPress.

Prérequis

Pour utiliser les widgets, votre thème doit être préalablement paramétré pour les accueillir. Lorsque vous choisissez un thème, vérifiez bien qu'il est "widgetisé". Si ce n'est pas le cas, il vous faudra alors insérer du code dans certains des fichiers de votre thème pour l'utiliser avec les widgets. Toutes les instructions pour rendre un thème "widgetisable" sont fournies au Chapitre 5. Cette section vous apprendra également ce qu'il faut regarder dans un thème pour savoir s'il est widgetisé ou pas.

Installation

Pour installer vos widgets, allez dans Apparence > Widgets. Sur la gauche, une série de widgets apparaissent (voir Figure 4.07).

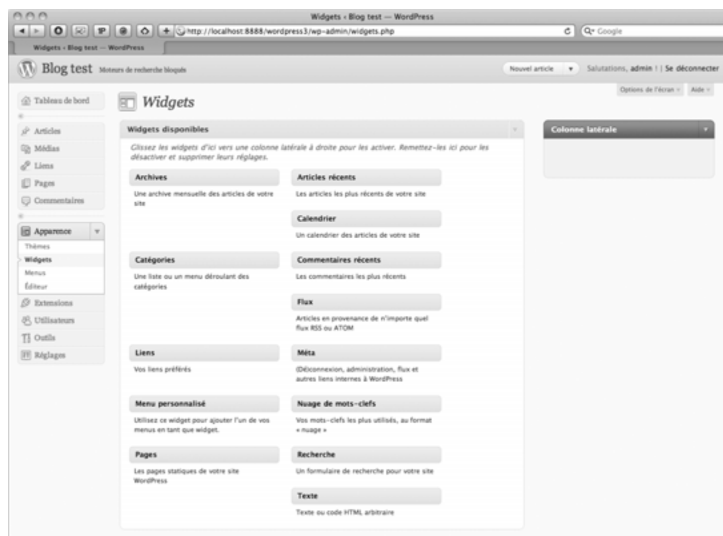


Figure 4.07
Widgets proposés par défaut.

À chaque ligne correspond un petit bloc qui, s'il est ajouté, viendra s'installer dans votre barre latérale.

Trouvez donc le widget qui vous intéresse et cliquez sur le lien Ajouter. À partir de là, ce widget va apparaître dans la colonne de droite intitulée Widgets actuels. Sur la droite du widget, vous trouverez un lien Modifier qui vous donnera accès à ses options (voir Figure 4.08).

En effet, si certains installent une fonction simple sur le blog, d'autres demandent un paramétrage de votre part.



Figure 4.08
Options d'un widget.

Enregistrez les modifications et allez sur la page d'accueil de votre blog pour voir le résultat. Vous pouvez constater que le bloc que vous avez choisi apparaît bien dans la colonne de droite, mais que les blocs présents précédemment ont disparu.

En effet, sur la plupart des thèmes WordPress, c'est l'ensemble de la colonne latérale qui est widgetisée. Dès qu'un widget est activé, le contenu qui n'est pas widgetisé disparaît.

L'un des avantages des widgets est de pouvoir gérer le contenu de votre colonne latérale sans avoir besoin de toucher au code du thème. Et cet avantage prend toute son envergure avec la possibilité d'intervertir les widgets.

En effet, vous pouvez changer le positionnement de chacun de vos widgets actuels grâce à la technique drag and drop qui est utilisée ici (voir Figure 4.09). Il suffit de les saisir avec la souris, de les déplacer et de les positionner comme vous le souhaitez, les uns en dessous des autres.



Figure 4.09
Le drag and drop des widgets.

N'oubliez pas d'enregistrer les modifications pour que les changements soient pris en compte.

Utilisation et gestion des extensions

Comme nous l'avons dit précédemment, les extensions sont des petites applications qui vont venir se greffer à votre blog pour lui apporter une fonctionnalité supplémentaire. Leur utilité est très diverse et peut aller de l'ajout d'un système de statistiques à la création d'un "site-map" pour votre blog, en passant par des outils permettant de visualiser des vidéos. Dans cette section, nous allons voir comment installer ces extensions, comment les mettre à jour et lesquelles vous apparaîtront rapidement indispensables.

Installation

Installation à partir de la base WordPress.org

WordPress permet à partir de la version 2.7 d'installer automatiquement des extensions en provenance de leur base de données, dont le site web est **http://wordpress.org/extend/plugins/**.

Depuis la page Extensions > Ajouter, vous avez accès à toutes ces extensions et vous pourrez les installer directement à partir de votre interface d'administration.

Sur cette page, vous avez tout d'abord toute une série d'onglets qui vous permettent de naviguer sur la base et de trouver notamment les extensions les plus populaires, les dernières mises à jour ou mises en ligne, ou encore celles qui font l'objet d'une mise en avant (voir Figure 4.10). C'est un progrès notable pour trouver ce qu'on cherche sans avoir besoin de parcourir le Web.



Figure 4.10
Onglets de navigation sur la base des plugins de WordPress.org.

Vous pouvez également faire une recherche sur cette base grâce au formulaire qui permet de chercher par terme, auteur ou étiquette (voir Figure 4.11).



Figure 4.11
Formulaire de recherche de plugins.

Une fois que vous avez trouvé l'extension que vous souhaitez utiliser, il vous suffit de cliquer sur le lien Installer, à droite dans la colonne Actions. Une nouvelle page s'ouvre alors au premier plan et vous donne les détails de l'extension directement en provenance du site web de WordPress (voir Figure 4.12).

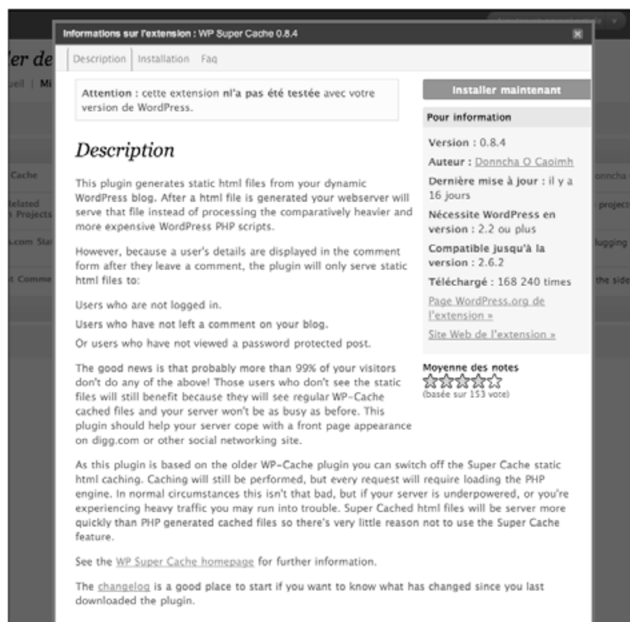


Figure 4.12

Détails de l'extension.

Validez l'installation en cliquant sur le bouton Installer maintenant, situé en haut à droite. La fenêtre se referme alors et WordPress procède à l'installation de l'extension.

Vous pouvez suivre le processus d'installation directement sur la page Extensions > Ajouter (voir Figure 4.13).



Figure 4.13

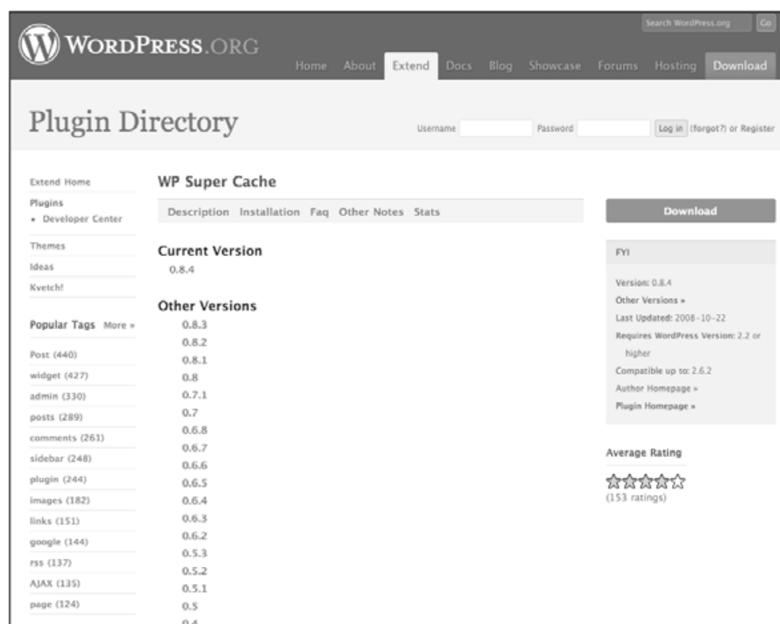
Installation d'une extension.

Lorsque l'extension est installée, WordPress vous propose de l'activer pour qu'elle soit opérationnelle. L'outil vous propose également d'aller sur la page des extensions sans activer l'extension (vous pouvez toujours le faire directement à partir de la page des extensions).

L'extension est maintenant activée, et vous pouvez l'utiliser. N'hésitez pas à aller sur la page du plugin sur le site **http://wordpress.org/extend/plugins/** pour avoir plus d'informations à ce sujet (voir Figure 4.14). Certaines extensions exigent des paramétrages supplémentaires pour être fonctionnelles.

Figure 4.14

Informations concernant l'utilisation d'une extension sur WordPress.org.



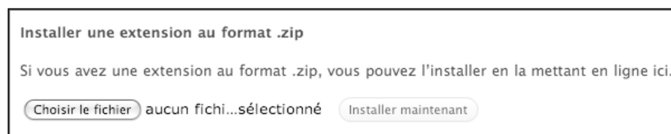
Installation en téléchargeant une extension à partir de l'interface d'administration

Toujours sur la même page Extensions > Ajouter, vous avez la possibilité de transférer les fichiers d'une extension depuis votre ordinateur directement dans le dossier dédié aux extensions sur votre hébergement. Il vous faut tout d'abord aller dans l'onglet Envoyer, en haut de la page, puis cliquer sur le bouton Choisir le fichier et aller sur votre disque dur pour trouver le ou les fichiers de l'extension que vous souhaitez installer.

Une fois que vous avez trouvé le fichier à transférer, cliquez sur le bouton Choisir de votre navigateur. Le fichier apparaît alors sur la page Extensions > Ajouter et vous n'avez plus qu'à cliquer sur le bouton Installer maintenant (voir Figure 4.15).

Figure 4.15

Installation d'une extension à partir d'un transfert de fichier.



Maintenant que l'extension est installée, il vous suffit de cliquer sur le bouton Activer ou d'aller sur la page Extensions > Installé pour activer votre extension.

Pour cela, vous devez descendre en bas de la page dans la liste des extensions inactives et l'activer en cliquant sur le lien situé à droite, sur la même ligne.

Installation en transférant une extension via FTP

Le processus est analogue à l'installation en téléchargeant l'extension à partir de l'interface d'administration, si ce n'est qu'ici vous allez transférer les fichiers de l'extension *via* votre logiciel FTP.

Déposez ces fichiers dans le dossier wp-content > plugins et allez ensuite sur la page Extensions > Installé. Comme pour les installations précédentes, vous devrez cliquer sur Activer pour que l'extension soit opérationnelle.

Mise à jour des extensions

Les extensions évoluent avec le temps et les différentes versions de WordPress. Il vous faudra donc les mettre à jour régulièrement. Cependant, vous n'avez pas besoin de rechercher ces mises à jour vous-mêmes, elles vous seront proposées automatiquement.

Les mises à jour à effectuer apparaîtront automatiquement au niveau du menu de navigation, au niveau de l'onglet Extensions. En allant sur la page Installées, les extensions qui devront être mises à jour seront accompagnées d'un petit bloc de texte vous invitant à mettre à jour l'extension de plusieurs manières possibles :

- Soit vous faites la mise à jour automatiquement, auquel cas cela se passera de la même manière que pour l'installation automatique d'une extension *via* l'interface d'administration.
- Soit vous téléchargez les fichiers et faites la mise à jour comme une installation classique d'une extension, à la différence qu'ici il faudra penser à désactiver l'extension avant de la mettre à jour et qu'il vous faudra tout simplement remplacer les fichiers existants sur votre hébergement pour mettre à jour votre extension (pensez à la réactiver par la suite).

Les extensions par défaut

Par défaut, WordPress propose deux extensions.

Akismet

Cette extension est sûrement l'une des plus importantes pour votre blog. C'est elle qui va le protéger du spam, le fameux "courrier" publicitaire poubelle. Sur un blog, le spam se manifeste souvent par le biais des commentaires. Si votre blog a du succès, vous pouvez en recevoir des centaines par jour, ce qui est impossible à gérer manuellement.

Si l'installation d'une extension reste très simple, celle d'Akismet est plus complexe. Vous allez devoir vous rendre sur un site pour obtenir une clé API qui vous permettra d'utiliser l'outil. Attention, Akismet est payant pour les entreprises.

Hello Dolly

C'est le premier plugin développé pour WordPress. Malgré tout, celui-ci est d'une utilité toute limitée puisqu'il ne propose que des passages de la fameuse chanson de Louis Armstrong, en haut à droite de chaque page de l'interface d'administration de votre blog. Cependant, Hello Dolly est souvent utilisé pour comprendre comment fonctionnent les extensions de WordPress. Nous nous en servirons comme modèle dans la dernière partie de l'ouvrage pour apprendre à créer une extension.

Les indispensables

À l'instar des thèmes, les plugins pour WordPress sont innombrables. Sur le site officiel de WordPress.org (http://wordpress.org/extend/plugins), ce ne sont pas moins de 2 700 plugins qui sont référencés. Et, dans tout ça, il n'est pas facile de faire son choix.

Ci-après, vous trouverez une liste complète des extensions, qui, si elles ne sont pas obligatoires, sont toutefois fortement recommandées.

Google XML Sitemaps (http://www.arnebrachhold.de/projects/wordpress-plugins/google-xml-sitemaps-generator/)

Cette extension, bien qu'elle soit quelque peu technique, est très importante si vous souhaitez indexer votre blog sur les moteurs de recherche. En effet, elle va créer ce qu'on appelle un "sitemap" ou plan du site en français, qui sera utilisé par Google et les autres moteurs de recherche pour indexer plus facilement les différentes pages de votre blog.

WordPress Database Backup (http://www.ilfilosofo.com/blog/wp-db-backup/)

Tout comme votre ordinateur, votre blog a parfois besoin d'une sauvegarde. Personne n'est à l'abri d'une panne de serveur, et il y a donc toujours un risque de perdre des données. Grâce à cette extension, vous allez pouvoir créer ou planifier des sauvegardes de votre base de données facilement.

Attention, ce genre d'outil ne sauvegarde pas les fichiers de votre blog, mais uniquement ceux de la base de données. C'est à vous de planifier régulièrement et manuellement une sauvegarde des fichiers de votre site.

Subscribe to comments (http://txfx.net/code/wordpress/subscribe-to-comments/)

Subscribe to comments permet à l'ensemble des personnes qui laissent des commentaires sur votre blog de s'abonner au fil de discussion de l'article qu'elles ont commenté. En clair, la personne laisse un commentaire et peut s'abonner pour recevoir un e-mail dès qu'un autre commentaire est posté.

Les blogs engendrent des discussions, dit-on souvent, et cette extension permet d'être tenu facilement au courant de ce qui se dit sur un article et d'intervenir une nouvelle fois si besoin est.

Simple Tags (http://wordpress.org/extend/plugins/simple-tags/)

Cette extension va vous permettre de mieux gérer l'ensemble de vos tags et de tirer profit de leur utilisation. Vous pourrez par exemple associer les tags de vos articles aux mots-clefs figurant dans l'en-tête de vos pages web. Vous pourrez également créer un nuage de tags paramétrables pour votre blog. Le plugin comprend bien d'autres options tout aussi intéressantes.

cforms II (http://www.deliciousdays.com/cforms-plugin)

Il y a de grandes chances que des lecteurs veuillent vous contacter *via* votre blog. Une bonne solution pour rester "joignable" est de proposer un formulaire de contact. Ainsi, un peu comme pour un commentaire, la personne peut laisser son nom, son adresse e-mail, ainsi qu'un message.

Similar Posts (http://rmarsh.com/plugins/similar-posts/)

Cette extension propose les articles du blog comparables à celui qui s'affiche. Vous trouverez souvent cette rubrique appelée Articles en relation sur de nombreux blogs (voir Figure 4.16).

Figure 4.16
Articles identiques.

Ce type d'extension vous demandera d'insérer un bout de code dans un ou plusieurs des fichiers du thème que vous utilisez.

Ressources web pour trouver des extensions

Contrairement aux thèmes, les ressources qui regroupent l'ensemble des extensions pour WordPress sont moins nombreuses. Cela dit, il existe tout de même de très bons sites qui vous permettront de trouver la perle rare.

WordPress Plugin Directory (http://wordpress.org/extend/plugins)

C'est la référence incontournable. Ce site regroupe l'ensemble des extensions qui existent pour WordPress. Si elle ne figure pas ici, il y a de grandes chances pour qu'elle ne soit plus

maintenue ou qu'elle ne soit pas de grande qualité. Le système de recherche et de notation est vraiment très utile pour trouver la meilleure extension.

Geekeries (http://www.geekeries.fr/)

Ce site est une vraie mine d'or pour trouver des extensions WordPress, dont une grande partie en français. À ne pas manquer.

Lester Chan (http://lesterchan.net/portfolio/programming/php/)

On compte beaucoup de développeurs d'extensions pour WordPress. Quelques-uns cependant sont très connus pour avoir créé certaines des extensions les plus utilisées. C'est le cas de Lester Chan qui a déjà créé une quinzaine d'extensions. N'hésitez pas à aller sur son site, vous y trouverez sûrement une ou deux extensions intéressantes pour votre blog. À noter que, depuis le début de l'année 2010, Lester ne met plus à jour ces extensions pour les nouvelles versions de WordPress. Quoi qu'il en soit, son "œuvre" reste à ce jour incontournable.

Alex King (http://alexking.org/projects/wordpress)

Alex King est lui aussi très connu pour ses plugins WordPress. Un incontournable qu'il ne faut absolument pas manquer.

Gestion des menus

Comme à chaque mise à jour, la version 3 de WordPress apporte son lot de nouveautés. Parmi celles-ci, une des plus importantes est probablement la nouvelle gestion des menus. Jusqu'à présent, pour créer un menu, il vous fallait créer des pages ou des catégories et mettre le bon code dans le thème pour voir apparaître l'un ou l'autre.

Avec WordPress 3, vous avez la possibilité de choisir ce que vous voulez voir apparaître dans votre menu, et l'ordre dans lequel vous voulez voir apparaître ces différents éléments.

Il y a cependant une chose importante à noter. Tous les thèmes ne permettent pas d'utiliser cette nouvelle fonctionnalité. En effet, il faut pour cela "autoriser" cette utilisation en ajoutant quelques lignes de code dans le thème que vous utilisez. Nous expliquons comment faire plus loin dans le livre, si vous souhaitez créer vous-même votre thème.

Dans le cas présent, nous allons utiliser le thème par défaut, Twenty Ten. Si vous allez sur la page de votre site, vous allez trouver un menu par défaut directement sous l'image de l'en-tête. Ce menu affiche deux éléments, Accueil et À propos. En créant un nouveau menu, vous allez remplacer ces deux éléments.

Ce qui est intéressant dans cette nouvelle gestion des menus, c'est que vous pouvez choisir différents types de liens à afficher dans votre menu.

Créez par exemple un nouveau menu que vous allez appeler Navigation (voir Figure 4.17).

Une fois ce menu créé, vous avez, dans la colonne de gauche, différentes informations qui vont vous permettre de paramétrer votre menu.

Figure 4.17
Création d'un menu.

Tout d'abord, vous avez la possibilité de créer plusieurs menus sur votre thème, à condition, bien entendu, que celui-ci le permette. Mais il arrive fréquemment de voir des thèmes qui proposent un menu avec les pages du site et un autre avec les catégories du site.

Ainsi, dans le premier bloc de la colonne de gauche, vous pouvez choisir l'emplacement de votre menu. Ici, on utilise le thème par défaut, il n'y a donc qu'un seul emplacement, Navigation (voir Figure 4.18).

Figure 4.18
Emplacement du menu.

En dessous, vous trouverez différents blocs qui vous permettront de déterminer chacun des éléments de votre menu (voir Figure 4.19).

- **Lien personnalisé.** Ici, vous pouvez choisir d'insérer un lien vers une page extérieure, quelle qu'elle soit.
- **Page.** Liez ici votre menu à une ou à plusieurs pages de votre blog.
- **Catégories.** Liez votre menu à une ou à plusieurs catégories de votre blog.

Figure 4.19
Différentes possibilités pour aménager votre menu.

Ainsi, votre menu créé et mis à jour, vous pourrez découvrir les nouveaux éléments directement sur votre blog. Vous pouvez également les modifier à n'importe quel moment.

Utilisation spécifique du thème par défaut, Twenty Ten

Lorsque vous ouvrez le menu Apparence et que vous utilisez le thème par défaut, Twenty Ten, différents menus dédiés à son utilisation s'affichent. C'est notamment le cas pour les sous-menus Background et Header.

Ces sous-menus sont dédiés à Twenty Ten. Cela veut dire qu'ils n'apparaîtront que si vous utilisez ce thème, et pas un autre.

Arrière-plan

Dans le sous-menu Arrière-plan, vous pouvez choisir l'arrière-plan de votre site. Cela peut être soit une image, soit une couleur.

Si vous avez une image que vous souhaiteriez utiliser comme fond, il vous suffit de charger sur votre site ce motif comme vous le propose WordPress (voir Figure 4.20).

Figure 4.20
Possibilités d'affichage pour l'arrière-plan.

Une fois l'image mise en ligne, elle apparaît directement sur la page avec différents paramètres à choisir (voir Figure 4.21).

Vous pouvez également choisir de changer la couleur de fond à la place de celle présente par défaut. Saisissez directement le code hexadécimal ou bien choisissez une couleur sur la palette qui vous est fournie en cliquant sur le lien.

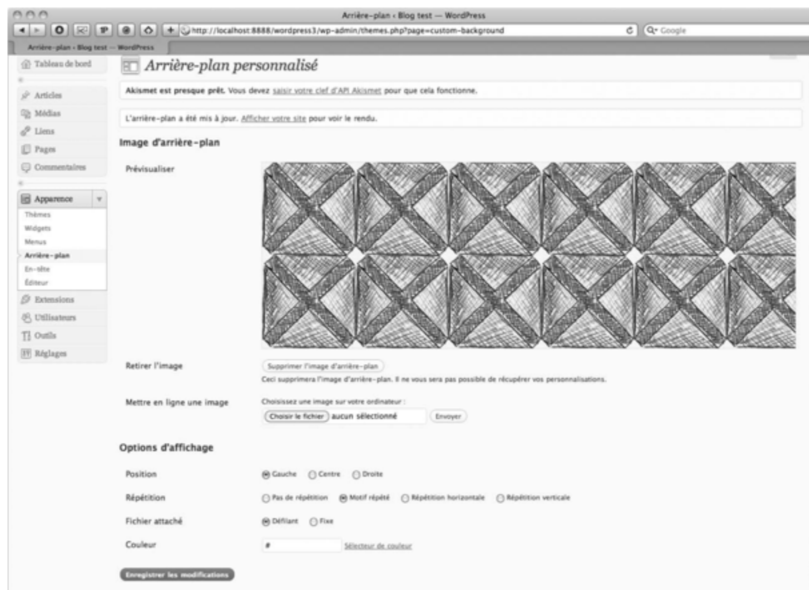


Figure 4.21
Possibilités d'affichage pour une image en arrière-plan.

En-tête

L'en-tête du thème Twenty Ten possède une image. *Via* le sous-menu En-tête, vous avez la possibilité de changer cette image. WordPress vous en propose quelques-unes mais vous pouvez également charger une image personnelle directement sur votre blog (voir Figure 4.22).

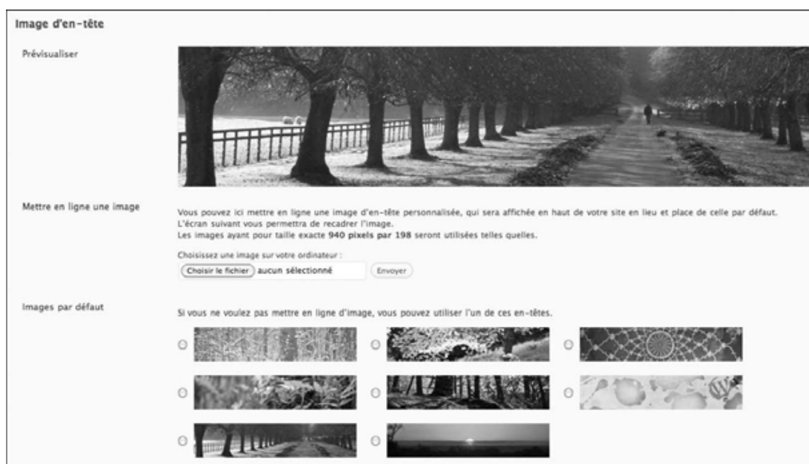


Figure 4.22
Onglet En-tête du thème Twenty Ten.

Partie



LE CAMPUS

WordPress côté développeur : concevoir un thème

5 Comprendre le fonctionnement d'un thème WordPress

Le thème est une partie incontournable de WordPress. C'est le thème qui donne son identité au blog, et c'est la première chose que voit le visiteur. Le thème habille le contenu du blog et prend en charge tous les aspects de mise en page et de design : polices, organisation, couleurs…

Dans ce chapitre dédié au développement d'un thème, vous apprendrez à mieux connaître tous les aspects d'un thème WordPress. Afin de ne pas sortir des sentiers battus, tous nos exemples de code seront basés sur le nouveau thème par défaut de WordPress, nommé Twenty Ten, et qui se trouve dans le dossier /wp-content/themes/twentyten de votre installation de WordPress.

Dans le chapitre suivant, vous partirez d'une page blanche et concevrez de A à Z un thème pour votre blog. Si vous êtes pressé, vous pouvez passer directement à ce chapitre.

Analyse de la structure d'un thème pour WordPress

Vous avez pu voir dans la partie précédente que les thèmes ont un dossier qui leur est propre au niveau de l'installation de WordPress : /wp-content/themes.

Par défaut, un thème est disponible : Twenty Ten. Dans le dossier du thème, on retrouve les différents fichiers liés au thème et nommés selon une convention précise :

- la feuille de style du blog : style.css ;
- les fichiers composant le thème, appelés "fichiers de modèle" : index.php, comments.php, header.php, footer.php, single.php, etc. ;
- un fichier pour les fonctions liées au thème, agissant comme une extension : functions.php.

Le thème par défaut, Twenty Ten, contient par ailleurs un sous-dossier, /images, où sont stockées toutes les images qu'il utilise, ainsi qu'un sous-dossier "languages".

Chaque fichier joue un rôle bien précis, dans une hiérarchie souple et cohérente.

La feuille de style

Le fichier style.css contient tous les aspects de mise en page du thème de votre blog, écrit à l'aide du langage CSS (*Cascading Style Sheet*). C'est ici que sont enregistrés notamment tous les choix relatifs à l'emplacement, la couleur et la typographie. C'est également ce fichier qui contient les informations que WordPress affichera au sujet du thème : nom du thème, version, nom de l'auteur…

Les fichiers de modèle

L'appellation "fichiers de modèle" (*template files*) regroupe tous les fichiers au format .php du thème, hormis les fichiers spéciaux comme functions.php.

Il existe deux types de fichiers de modèle :

- **Les fichiers qui vont composer le thème.** Reliés entre eux, ils vont créer et afficher la page web.
- **Les fichiers qui vont afficher les différentes pages du blog.**

Seuls deux fichiers sont obligatoires pour qu'un thème puisse être considéré comme valide par WordPress : index.php pour le code PHP, et style.css pour la mise en forme du blog. Notez qu'il suffit que le fichier style.css soit absent ou mal formaté pour que WordPress ne prenne pas le thème en compte.

Avec un thème qui n'utiliserait que ces deux fichiers, le fichier index.php doit contenir l'ensemble des appels PHP qui récupèrent et agencent les informations, pour les afficher sur le blog en fonction des requêtes du visiteur : en-tête, pied de page, textes, commentaires, colonne latérale, moteur de recherche… Cependant, si vous souhaitez personnaliser votre thème, il est préférable de le segmenter en différents fichiers qui correspondront notamment à différentes parties du blog : header.php pour l'en-tête, footer.php pour le pied de page, sidebar.php pour la colonne… Nous verrons très bientôt les différents noms de fichier possibles.

Le fichier functions.php

Ce fichier agit un peu comme une extension de WordPress et n'est pas obligatoire. Grâce à ce fichier, vous pouvez présenter différentes options et paramètres du thème à l'utilisateur *via* l'interface d'administration de WordPress, sans pour autant faire des modifications directement dans le thème. Il est ainsi possible, avec un peu de travail et de prévoyance, de laisser l'utilisateur choisir le nombre de colonnes, l'image en haut du blog, la taille des textes, etc. Les possibilités sont infinies.

Le dossier images

Ce dossier contient toutes les images qui servent à la présentation de votre blog. Il n'est pas obligatoire, mais à partir de trois images, il est préférable de les ranger dans un dossier à part. Le nom de ce dossier n'est pas non plus fixé, et vous pouvez le baptiser à votre gré : images, img, pix…

Notez bien que ce n'est pas dans ce dossier que vous devez ranger les images que vous avez l'intention d'utiliser dans vos articles. En effet, si vous les insérez ici, elles disparaîtront le jour où vous changerez de thème.

Structure et convention de nomenclature

Ce ne sont là que quelques exemples de la structure "classique" que vous trouverez sur de nombreux thèmes WordPress. Cette structure se présente le plus souvent comme suit :

- **header.php.** Affiche les informations de l'en-tête du blog.
- **index.php.** Affiche le contenu du blog.
- **sidebar.php.** Affiche la colonne latérale du blog.
- **footer.php.** Affiche le pied de page du blog.

Notez que trois fichiers ont été extraits du header.php de base, pour gérer respectivement trois zones du thème indépendantes du contenu : l'en-tête, le pied de page et la colonne latérale.

Voici un exemple d'organisation simple de la page d'accueil d'un blog WordPress utilisant le thème par défaut(voir Figure 5.01).

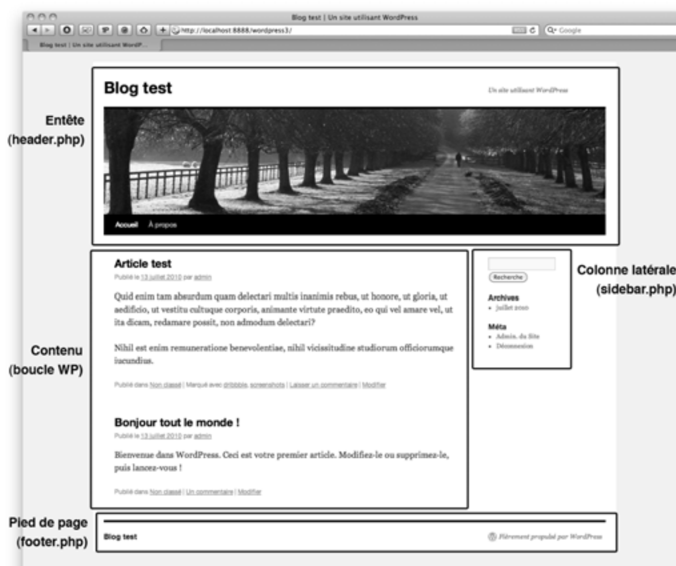


Figure 5.01
Page d'accueil.

Lors de l'affichage d'un seul article du blog, WordPress fera appel aux fichiers suivants, liés entre eux par des appels PHP et les conventions de WordPress :

- **single.php.** Affiche le contenu de l'article.
- **header.php.** Affiche l'en-tête du blog.
- **sidebar.php.** Affiche la colonne latérale du blog.
- **footer.php.** Affiche le pied de page du blog.
- **comments.php.** Affiche les commentaires et le formulaire de commentaire.

En séparant le thème en plusieurs fichiers, chacun s'occupant d'une partie précise de l'affichage, on simplifie les fichiers, et leur développement devient bien plus aisé. Le fichier HTML final est construit à l'aide d'appel de méthodes PHP internes à WordPress, qui assument le chargement des fichiers nécessaires selon leurs noms et la hiérarchie existante, ce que nous verrons dans la section qui suit.

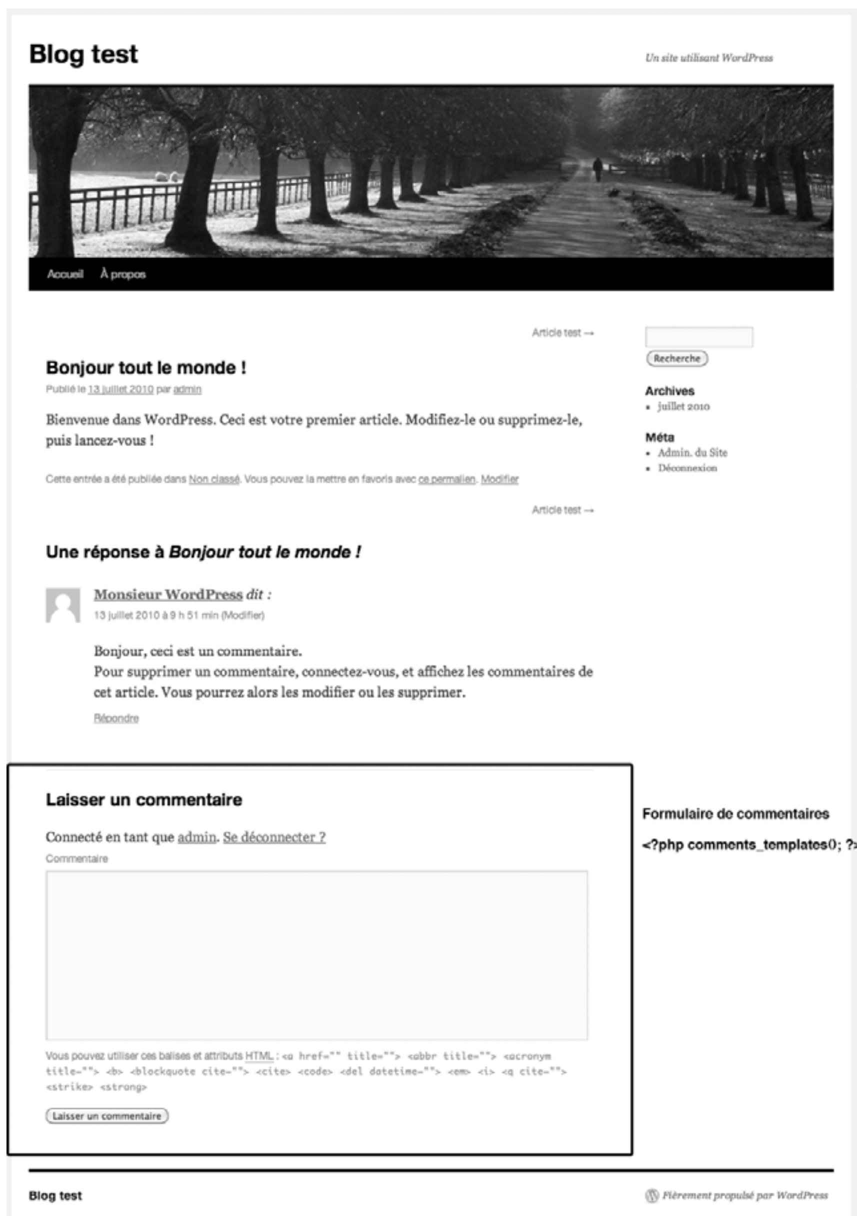


Figure 5.02
Page de l'article avec mise en relief des commentaires.

Sur les autres types de pages, comme les pages de catégories, de labels (tags), d'archives ou encore de recherche, nous avons une combinaison tournant autour de ces fichiers :

- index.php, category.php, 404.php ou search.php : fichiers de modèle spécifiques à un affichage ;
- header.php ;
- sidebar.php ;
- footer.php.

Le thème reprend généralement la même organisation que pour la page d'accueil, mais il est également possible de créer un fichier typique pour chacune de ces pages, comme category.php pour les pages de catégories.

Différents fichiers pour différentes pages

Nous venons de voir qu'il est préférable de scinder une page web en différents fichiers et de les appeler par des fonctions PHP de WordPress. Nous avons également abordé la structure HTML de base qui sera utilisée pour créer un thème WordPress. Celle-ci sera à déployer sur les différents fichiers qui afficheront différents types de pages du blog. Il reste à définir les fichiers que nous pouvons utiliser.

Voici la liste complète des fichiers permettant de gérer séparément des parties du blog :

- **single.php.** Pour afficher un article seul.
- **page.php.** Pour afficher une page seule (on peut également utiliser pagename.php qui affichera directement le nom de la page).
- **home.php.** Pour afficher une page d'accueil spécifique.
- **tag.php.** Pour afficher les pages de tags (on peut aussi employer tag-slug.php qui affichera les articles correspondant à un tag en ayant son nom directement dans l'URL sous la forme "tag-wordpress.php" par exemple, avec ici "wordpress" comme nom de tag).
- **archive.php.** Pour afficher les archives (par catégorie, par label, par auteur, etc.).
- **category.php.** Pour afficher une catégorie seule. Il peut être utilisé sous une déclinaison encore plus précise, category-X.php, pour gérer l'affichage des articles de la catégorie dont le numéro d'identifiant est X (par exemple, category-7-.php).
- **search.php.** Pour afficher les résultats d'une recherche.
- **404.php.** Pour afficher la page d'erreur, quand un article n'est pas disponible par exemple (ce qu'on appelle une erreur 404).
- **[nom de la page].php.** Pour afficher une page précise, à partir de son permalien. Par exemple, si nous voulons un affichage spécifique pour la page À propos, nous utiliserons a-propos.php.
- **author.php.** Pour afficher une page présentant les données liées à un auteur en particulier (articles, pages, liens, autres informations).
- **date.php.** Pour afficher les articles par date précise (année, mois ou jour).

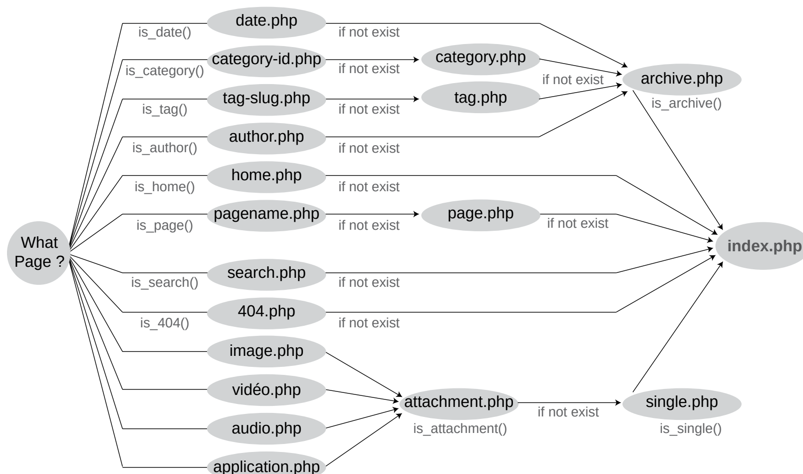
- **image.php, video.php, audio.php et application.php.** Pour afficher les pages correspondant à ces différents types de médias qui ont leur propre URL. En créant un marqueur de modèle particulier, vous pourrez générer des pages spécifiques pour chacun d'eux.

Hiérarchie des différents types de pages

Les noms de fichiers présentés ci-dessus servent pour afficher les différentes vues disponibles du blog, sans être pour autant obligatoires. En effet, les modèles de WordPress fonctionnent autour d'une hiérarchie précise, et si un fichier n'existe pas dans le thème, WordPress ira chercher le fichier supérieur dans cette hiérarchie pour afficher le contenu.

La Figure 5.03 représente l'organigramme de hiérarchie des modèles dans WordPress.

Figure 5.03
Hiérarchie des modèles.



Prenons un exemple pour illustrer la logique de cette hiérarchie. Un visiteur passe sur un blog et clique sur le lien pour voir tous les articles de l'année 2008. Cette URL a la forme suivante : *http://www.monblog.com/2008*.

Si nous reprenons les règles de nommage des fichiers citées ci-dessus, la page devrait s'afficher à l'aide du fichier date.php. Or le thème employé ne dispose pas d'un fichier de ce nom. En utilisant la hiérarchie des modèles, nous pouvons voir que dans ce cas WordPress ira trouver le fichier archive.php pour afficher les articles de l'année 2008. Et s'il n'existe pas non plus, il remontera jusqu'au fichier index.php qui, lui, est obligatoire.

Il en est de même pour l'ensemble des fichiers présentés plus haut. Ils suivent une hiérarchie qui permettra toujours d'afficher une page web avec le contenu demandé, mais peut-être pas sous une forme aussi affinée que si le thème avait été créé avec chaque fichier propre à chaque vue du blog.

Les fichiers propres à chaque thème

Les fichiers qui composent cette hiérarchie sont les seuls que vous pouvez utiliser pour afficher les pages du blog. Vous ne devez pas en créer d'autres, ils ne seraient pas reconnus automatiquement par WordPress.

Par contre, pour les fichiers qui vont composer les différentes parties de ces pages, il en va tout autrement. Vous pouvez très bien créer un fichier spécifique pour une partie précise de votre blog, par exemple intégrer le fichier pubs.php dans une partie de la colonne latérale pubs.php, afin de gérer votre publicité à part.

Pour le thème Twenty Ten, il existe plusieurs autres fichiers qui lui sont spécifiques et qui ont tous une fonction particulière.

- **loop.php.** Pour le thème Twenty Ten, les développeurs ont décidé de séparer la boucle WordPress pour la mettre dans un fichier séparé. La boucle est ce qui permet d'afficher les articles du site en fonction de la requête que vous aurez faite. Nous y reviendrons dans le détail plus loin dans le chapitre.
- **one-column-page.php.** Ce fichier a été créé dans le cas où vous souhaiteriez avoir un site sans colonne latérale.
- **sidebar-footer.php.** Ce fichier correspond à la zone widgetisée qui est dans le pied de page du site.
- **editor-style.css.** Cette feuille de style est celle utilisée pour l'éditeur TinyMCS, que vous trouvez sur les pages de création d'un article ou d'une page.
- **editor-style-rtl.css et rtl.css.** Feuilles de style utilisées pour les langues qui se lisent de droite à gauche, comme l'arabe ou l'hébreu.

Structure HTML d'un thème WordPress

Maintenant que nous avons vu les différents fichiers possibles et l'utilité de chacun, il est temps de nous plonger dans le contenu de ces fichiers. Celui-ci mélange balises HTML (ou plutôt XHTML pour les puristes) et code PHP pour gérer les appels aux fonctions de WordPress, la mise en place de conditions et de boucles, bref tout ce qui au final générera un fichier HTML avec toutes les données voulues par le visiteur. Il est donc préférable que vous connaissiez déjà les langages XHTML et CSS avant d'aller plus loin… Le PHP serait un plus, bien que vous puissiez très bien comprendre le fonctionnement d'un thème WordPress sans pour autant avoir de notions en PHP.

Voyons donc ce que va donner le découpage d'un thème au niveau HTML. Nous allons prendre ici un exemple simple, qui permettra de bien comprendre l'architecture d'un thème WordPress.

Nous partons avec un thème dont la structure est axée autour de quatre parties : l'en-tête, le contenu, la colonne latérale et le pied de page. Pour cet exemple, nous allons limiter les fichiers au strict minimum : index.php et style.css.

La structure HTML fondamentale du fichier index.php va donc donner :

```
<body>
  <div id="page">
    <div id="header"></div>
    <div id="content"></div>
    <div id="sidebar"></div>
    <div id="footer"></div>
  </div><!-- end page -->
</body>
```

Notez que nous ne mentionnons ici que le contenu de la balise `<body>`. Pour un thème complet, il faudra bien entendu indiquer le doctype, la balise `<html>`, les métadonnées, etc. : tout ce qui fait un fichier XHTML valide.

Dans ces différentes balises viennent s'intégrer les fameux "template tags" de WordPress, ces marqueurs de modèle qui se chargeront d'appeler les fichiers concernés pour les afficher, selon la hiérarchie établie :

```
<body>
  <div id="page">
    <div id="header"><?php get_header(); ?></div>
    <div id="content"></div>
    <div id="sidebar"><?php get_sidebar(); ?></div>
    <div id="footer"><?php get_footer(); ?></div>
  </div><!-- end page -->

</body>
```

Ce schéma sera à répéter sur les différents fichiers qui afficheront le blog, tels que les pages d'articles, de catégories, de tags, etc.

Présentation des marqueurs de modèle

Nous avons abordé cette notion de template tags plusieurs fois auparavant. Les template tags sont traduits en français par "marqueurs de modèle", "balises de gabarit", "balises de modèle" ou "étiquettes des fichiers du blog".

Dans les faits, ces marqueurs sont de courtes fonctions PHP qui vont appeler d'autres fichiers à l'endroit où ils ont été insérés, ou charger du contenu. Ils sont extrêmement utiles et très nombreux sur WordPress. Vous en trouverez une liste complète à cette adresse : **http://codex.wordpress.org/Template_Tags**.

Ces fonctions PHP ont été créées spécialement pour faciliter le fonctionnement et le paramétrage d'un thème. C'est là une des forces de WordPress, qui en fait la plate-forme de blogs préférée des designers et des développeurs.

Par exemple, le marqueur `get_header()` appelle simplement le fichier header.php et inclut son contenu tel quel dans le code HTML du fichier. C'est un marqueur très facile à retenir et à manipuler, et il en existe de nombreux autres qui facilitent la vie du créateur de thèmes de la même manière.

En ce qui concerne les fichiers propres au thème, dont nous avons parlé un peu plus haut, il n'est pas possible d'utiliser de marqueurs prédéfinis par WordPress, étant donné que ce

fichier n'est pas prévu par défaut. C'est pourquoi les développeurs de WordPress ont créé une constante PHP qui vous permettra d'appeler n'importe quel fichier qui ne peut pas être appelé par un marqueur "classique", sans devoir préciser l'URL absolue du fichier, ni craindre d'utiliser une URL relative qui ne fonctionnerait pas dans certains cas :

```
<?php include(TEMPLATEPATH . '/nomdufichier.php'); ?>
```

`TEMPLATEPATH` est une constante qui correspond au dossier du thème, grâce à laquelle la fonction `include()` de PHP peut chercher le fichier "nomdufichier.php". De cette façon, le contenu de ce fichier s'affichera là où vous avez inséré ce bout de code, de la même manière que s'il s'agissait d'un vrai marqueur de modèle.

Le contenu dans la page web

Nous avons vu quels types de fichiers utiliser pour créer une page web. Il est également important de comprendre le contenu de chacun de ces fichiers. Dans la section qui suit, nous allons prendre un exemple simple, la page d'accueil du thèmeKubrick, ancien thème par défaut de WordPress, en présenter chacune des parties, et voir quelles informations s'affichent et quels marqueurs de modèle elles utilisent.

L'en-tête (header.php)

L'en-tête d'un blog WordPress comprend toutes les informations utiles et nécessaires au navigateur pour afficher votre blog. Ces informations sont obligatoires pour toute page web et sont comprises dans la balise `<head>`.

Nous nous intéressons ici au contenu de l'en-tête, donc à ce qui va être utilisé par le navigateur. En regardant l'en-tête du thème Twenty Ten tel qu'il s'affiche dans le navigateur (voir Figure 5.04), trois informations liées au blog ressortent : le titre du blog, sa description et le menu de navigation. Pour ce qui est du titre et de sa description, les deux informations sont créées automatiquement par un seul marqueur, grâce à deux de ses variantes.



Figure 5.04
En-tête default avec les template tags.

Le titre du blog utilise le template tag `<?php bloginfo('name'); ?>`. WordPress va chercher dans les informations du blog (`bloginfo`) le nom de celui-ci (`name`). Dans le même esprit, la description, quant à elle, utilisera le même marqueur de modèle, mais ira chercher la description du blog : `<?php bloginfo('description'); ?>`.

Ces informations sont contenues dans la base de données, et ce marqueur est appelé de nombreuses fois sous différentes variantes depuis le fichier header.php. Pour que ces informations soient lisibles par le visiteur, les deux variantes décrites ci-dessus sont appelées dans le `<body>` du fichier.

Pour ce qui est du menu de navigation, dont nous avons déjà parlé au chapitre précédent, c'est le template tag `<?php wp_nav_menu (); >` qui sera utilisé.

La colonne latérale (sidebar.php)

Le nombre d'informations affichées par défaut dans la colonne latérale peut être plus grand. On y trouve notamment un formulaire de recherche, les archives et les informations méta. Chacune de ces informations ou listes s'affiche par le biais d'un marqueur de modèle.

N'oubliez pas ici que ces informations disparaîtront dès que vous utiliserez les widgets. Le thème Twenty Ten est widgetisé mais affichera les informations dont nous venons de parler tant qu'aucun widget ne sera activé.

Voici donc une image reprenant la structure de la colonne latérale du thème default avec les différents marqueurs utilisés (voir Figure 5.05).

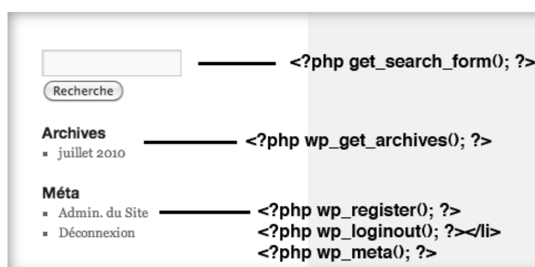


Figure 5.05
Sidebar + template tags.

Le pied de page (footer.php)

Le pied de page est généralement utilisé pour indiquer le nom du créateur du thème et renvoyer vers le site officiel de WordPress.

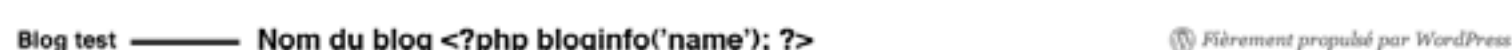


Figure 5.06
Footer + template tags.

Le thème utilise à nouveau le marqueur `bloginfo()` pour aller chercher le titre du blog (`'name'`). Le reste des informations est généré sans l'utilisation de marqueurs. Si vous souhaitez en modifier le contenu, il faudra le faire directement depuis le fichier footer.php.

Spécificité de la page d'article

Le fichier qui affiche un unique article du blog va suivre la même organisation. Cependant, il y a une grande différence par rapport à la page d'accueil : elle va afficher les commentaires liés à l'article. Pour cela, nous allons utiliser un fichier séparé, appelé comments.php. Ce fichier n'appartenant pas à la hiérarchie WordPress, nous devons l'appeler à partir d'un fichier qui en fait partie, en l'occurrence single.php.

Si comments.php ne fait pas partie de la hiérarchie, WordPress dispose tout de même d'un marqueur spécifique pour l'appeler : `comments_template()`. Celui-ci se charge de récupérer le fichier nommé comments.php et de l'inclure dans le fichier courant.

Dans le fichier single.php, on appellera ce marqueur de la même manière qu'une fonction PHP :

```
<?php comments_template(); ?>
```

Ce marqueur spécifique peut s'appliquer pour les pages statiques du blog, dans le cas où on souhaiterait y afficher des commentaires.

La boucle WordPress

Nous venons d'évoquer tous les fichiers qui vont pouvoir afficher les différentes parties du blog, sans toutefois aborder le contenu. Celui-ci est pourtant l'élément central du blog : il va se retrouver sur chaque page du blog.

Ce contenu est chargé et mis en place par ce qu'on appelle la boucle WordPress – the loop en anglais. Cette boucle met en place les conditions pour afficher les articles en fonction des requêtes du visiteur. Elle décide quoi afficher et comment l'afficher pour une URL donnée.

La Figure 5.07 reprend tout ce qui se trouve entre ces deux lignes et qui explique succinctement comment fonctionne la boucle WordPress.

Figure 5.07
Exemple de boucle WordPress.

Fonctionnement de la boucle WordPress

```
<?php if (have_posts()) : ?>    (Est-ce qu'il y a des articles à afficher ? - Ouverture de la boucle WordPress)
            <?php while (have_posts()) : the_post(); ?>   (dans le cas où il y en a, on les affiche...)

      <div class="post" id="post-<?php the_ID(); ?>">
            <h2><a href="<?php the_permalink() ?>" rel="bookmark" title="Lien permanent vers <?php the_title_attribute(); ?>"><?php the_title(); ?></a></h2>
            <small><?php the_time('j F Y') ?> <!-- par <?php the_author() ?> --></small>
            <div class="entry">
                  <?php the_excerpt('Lire le reste de cet article &raquo;'); ?>
            </div>
            <p class="postmetadata"><?php the_tags('Tags: ', ', ', '<br />'); ?> Publié dans
            <?php the_category(', ') ?> | <?php edit_post_link('Modifier', '', ' | '); ?>
            <?php comments_popup_link('Aucun commentaire »', '1 commentaire »', '% commentaires »', 'comments-link', 'Les commentaires sont fermés'); ?></p>
      </div>

       <?php endwhile; ?>  (On ferme l'affichage des articles...)

       <?php else : ?>     (s'il n' y a pas d'article à afficher...)

            <h2 class="center">Introuvable</h2>
            <p class="center">Désolé, mais vous cherchez quelque chose qui ne se trouve pas ici .</p>
            <?php include (TEMPLATEPATH . "/searchform.php"); ?>

       <?php endif; ?>(fermeture de la boucle WordPress)
```

D'un point de vue pratique, tout se passe de la manière suivante :

1. Le visiteur crée une requête sur le blog. Cette requête peut être aussi simple que l'affichage de la page d'accueil (la boucle affiche simplement les derniers articles) ou aussi complexe que l'utilisation du formulaire de recherche (la boucle doit afficher les articles contenant les mots saisis).
2. Une fois la requête lancée, la boucle vérifie s'il y a des articles à afficher (`<?php if (have_posts()): ?>`).

   Si oui, WordPress doit afficher ces articles un à un à l'aide du code précisé (`<?php while (have_posts()):the_post(); ?>`). Une fois le nombre d'articles voulu affiché, la boucle principale se termine (`<?php endwhile; ?>`).

   Si non (`<?php else : ?>`), WordPress doit en informer le visiteur avec un message et idéalement un moteur de recherche.
3. Lorsque tous ces traitements ont abouti, la boucle se termine (`<?php endif; ?>`).

Voyons maintenant dans le détail ce que va afficher WordPress, et surtout comment il va l'afficher. Nous allons prendre ligne par ligne et détailler le contenu de la boucle :

```
<div class="post" id="post-<?php the_ID(); ?>">
```

Dès le départ, il va personnaliser chaque article affiché séparément grâce au tag `the_ID`, qui renvoie le numéro unique de l'article. Cela peut être très utile dans le cas où vous souhaiteriez donner un style particulier à certains articles précis.

```
<h2><a href="<?php the_permalink(); ?>" rel="bookmark" title="Lien permanent
vers <?php the_title_attribute(); ?>"><?php the_title(); ?></a></h2>
```

La balise `<h2>` signifie qu'il va y avoir un titre ici. Ce titre s'affiche au sein d'un lien unique, celui de l'article. Pour ce faire, WordPress utilise trois marqueurs :

- `the_permalink()` pour obtenir l'URL unique de l'article ;
- `the_title()` pour obtenir le titre de l'article ;
- `the_title_attribute()` pour obtenir le titre de l'article, sous une forme "nettoyée" et donc appropriée, au sein de l'attribut title de la balise.

```
<small><?php the_time('j F Y'); ?> <!-- by <?php the_author(); ?> --></small>
```

En dessous du titre, on retrouve quelques informations, appelées "métadonnées de l'article". Elles affichent notamment la date et l'auteur par le biais de deux marqueurs :

- `the_time()` pour obtenir la date, selon le format de la fonction `date()` de PHP ;
- `the_author()` pour obtenir le nom public de l'auteur de l'article affiché.

```
<?php the_content('Lire le reste de l'article &raquo;'); ?>
```

Ensuite, le contenu même de l'article s'affiche. Il est possible d'utiliser deux marqueurs :

- `the_content()` pour obtenir l'article en entier ou jusqu'à la balise `[more]` de l'article. C'est le choix dans notre exemple.
- `the_excerpt()` pour obtenir une partie de l'article : soit l'extrait tel qu'il est défini explicitement par le blogueur, soit les 55 premiers mots de l'article. La balise `[more]` n'a pas d'influence ici.

Le marqueur `the_content()` peut par ailleurs recevoir en argument la phrase à afficher dans le cas où l'article contient une balise `[more]`. Dans notre exemple, le message "Lire le reste de l'article" apparaîtra sur la page et invitera le lecteur à lire le reste en cliquant sur le permalien de l'article.

```
<p class="postmetadata"><?php the_tags('Tags: ', ', ', '<br />'); ?> Publié
dans <?php the_category(', '); ?> | <?php edit_post_link('Modifier', '', ' |
'); ?>  <?php comments_popup_link('Aucun commentaire"', '1 commentaire"', '%
commentaires"', 'comments-link', 'Les commentaires sont fermés'); ?></p>
```

Enfin, on affiche le reste des métadonnées de l'article, chacune avec un marqueur dédié :

- `the_tags()` pour obtenir les labels donnés à l'article.
- `the_category()` pour obtenir les catégories où l'article a été placé.
- `edit_post_link()` pour afficher un lien permettant de modifier l'article. Ce lien n'est visible qu'à une personne habilitée à modifier les articles du blog.
- `comments_popup_link()` pour afficher un lien vers les commentaires (donc, dans les faits, un lien vers l'article lui-même).

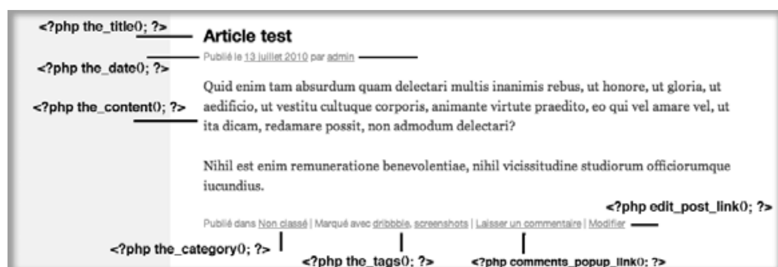
Ces marqueurs, on le voit, peuvent recevoir des paramètres. Les voici, par ordre d'apparition :

- `the_tags('Tags: ', ', ', '<br />')` : texte à afficher avant les labels ; séparateur à placer entre chaque label ; texte à afficher après les labels.
- `the_category(', ')` : séparateur à placer entre chaque catégorie.
- `edit_post_link('Modifier', '', ' | ')` : texte du lien ; texte à afficher avant le lien ; texte à afficher après le lien.
- `comments_popup_link('Aucun commentaire"', '1 commentaire"', '% commentaires"', 'comments-link', 'Les commentaires sont fermés')` : chacun des trois premiers paramètres correspond à un état des commentaires de l'article et permet donc de préciser la phrase à afficher ; le quatrième permet de préciser la classe CSS à appliquer au lien ; le cinquième et dernier paramètre correspond également à un état des commentaires et donc à la phrase à afficher.

Ci-après, un exemple de ce que vous allez obtenir comme résultat visuel avec les informations que nous venons de voir (voir Figure 5.08).

Figure 5.08

Article de la page d'accueil default.



6 Créer son propre thème WordPress

Dans ce chapitre, vous allez réaliser un thème de A à Z. Vous suivrez un processus de création simple, qui va du croquis au thème final en passant par différentes étapes, telles que la conception des fichiers nécessaires, la réalisation d'une première maquette et son intégration.

Pour pouvoir suivre et comprendre les différentes étapes de création d'un thème, il vous faudra tout de même quelques connaissances techniques, notamment en HTML, PHP et CSS. Le thème réalisé ici restera très simple, mais vous permettra de bien comprendre l'architecture générale du système de thème, et vous aidera à mettre votre créativité au service de WordPress.

La création du thème se passera en trois grandes étapes :

1. Vous allez créer les différents fichiers du thème qui vont permettre d'afficher le contenu. Il s'agira ici de mettre en pratique tout ce qui a été vu au chapitre précédent et de créer ces différents fichiers par vous-même.
2. Ensuite, vous passerez à une partie plus créative avec la réalisation d'un croquis et d'une première maquette pour avoir un aperçu visuel de votre thème.
3. Enfin, vous réaliserez l'intégration de votre thème et ferez les derniers réglages pour mettre le tout en ligne !

Pourquoi créer son propre thème ?

Avant d'entrer dans le vif du sujet et passer de longues heures à créer ce thème, il est important de répondre à cette question : pourquoi vouloir créer un thème pour WordPress ? En effet, il existe déjà beaucoup de thèmes, un grand nombre étant de très bonne qualité. Qui plus est, s'ils ne vous conviennent pas entièrement, vous avez toujours la possibilité de les modifier. Créer un thème de A à Z demande beaucoup de temps et surtout des notions techniques en PHP, HTML et CSS.

Cela dit, avoir votre propre thème, que vous aurez conçu à partir de rien, peut présenter de nombreux avantages. D'un point de vue technique :

- Vous concevez entièrement le thème. Vous connaissez donc chaque ligne de code et chaque style utilisé.
- Vous pouvez modifier, mettre à jour progressivement, ou encore optimiser votre thème. Cela représente un gain de temps considérable par rapport à un thème conçu par quelqu'un d'autre.
- Vous pouvez le créer selon vos propres méthodes.
- Vous saurez répondre aux problèmes plus rapidement.

D'un point de vue esthétique :

- Vous concevez votre propre design ; vous aurez donc un thème unique plutôt qu'un blog qui ressemble à tant d'autres.
- Vous disposez des fichiers originaux des images ; vous serez donc plus facilement en mesure de modifier tous les aspects graphiques comme bon vous semble, ce qui n'est pas toujours le cas avec les thèmes utilisant des images.

Plus prosaïquement, créer un thème peut être également un très bon outil de communication ! Si vous êtes graphiste ou webdesigner et si vous souhaitez améliorer votre visibilité sur le Web, voire simplement faire parler de vous, créer et diffuser un thème pour WordPress peut être une très bonne vitrine pour votre activité.

Enfin, c'est un excellent exercice pour mieux comprendre le fonctionnement de WordPress et se familiariser avec les langages qu'il utilise.

Création de la structure de base

Dans cette première partie, nous allons créer l'ensemble des fichiers qui vont composer votre thème. Nous explorerons ici la structure de base : si votre thème sera bien "utilisable" à la fin de cette première partie, il ne disposera pas cependant de graphiques ou de mise en page, car nous n'abordons pas encore la feuille de style.

Installer WordPress en local sur votre ordinateur

Pour réaliser votre thème, il vous faut une structure de test, ne serait-ce que pour pouvoir apprécier directement l'évolution de votre travail sans pour autant devoir l'utiliser sur votre blog public.

Deux solutions sont possibles : installer une seconde instance de WordPress sur votre hébergement, qui vous servira de blog de test pour tous vos développements personnels, ou installer WordPress en local, sur votre ordinateur.

Cette seconde solution a énormément d'avantages, comme la possibilité de travailler sur votre thème sans avoir besoin d'être connecté à Internet, c'est-à-dire sans devoir faire des allers-retours incessants entre votre éditeur pour l'écriture du code, votre client FTP pour la mise en ligne des fichiers modifiés, et votre navigateur pour vérifier l'impact de vos modifications sur le thème. Avec un serveur local, vous pourrez tout tester simplement en pointant votre navigateur vers un dossier de votre ordinateur, et une fois que le thème sera conçu et prêt à être mis en ligne, vous n'aurez qu'à transférer son dossier sur votre hébergement.

Installer WordPress localement ne se fait pas automatiquement : il faut utiliser ce qu'on appelle un package xAMP. Ce sigle fait référence aux trois serveurs nécessaires pour faire fonctionner un site dynamique : Apache pour le serveur web, MySQL pour le serveur de base de données, et PHP pour le langage de script. Le "x" est généralement remplacé par le système d'exploitation visé : LAMP pour Linux, WAMP pour Windows, et MAMP pour OS X (Macintosh). Un package xAMP facilite grandement la mise en place d'un serveur web local fonctionnel, et c'est la solution que nous vous présentons ici.

Deux des logiciels les plus connus sont XAMPP, qui existe pour plusieurs plates-formes (et peut fonctionner avec Perl, d'où le second "P"), et MAMP, qui lui ne fonctionne que sur OS X. Ils sont téléchargeables respectivement aux adresses suivantes : **http://www.apachefriends.org/fr/xampp.html** et **http://www.mamp.info/en/index.php**. Deux autres packages très utilisés sont WampServer/WAMP5 et l'ancêtre EasyPHP, tous deux français. Tous ces packages sont gratuits.

Installer XAMPP sur Windows

L'installation de XAMPP suit un cheminement très simple :

1. Téléchargez le package XAMPP pour Windows depuis le site de XAMPP (voir Figure 6.01). Prenez soin de choisir celui qui est marqué "Installer" plutôt que "ZIP" ou "EXE".

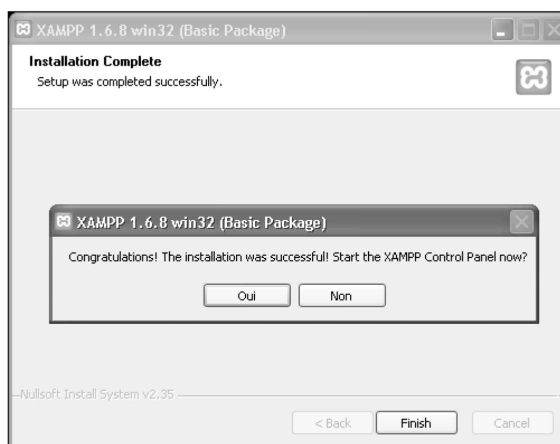
Figure 6.01
Téléchargement de XAMPP.

XAMPP pour Windows 1.6.8, 28.09.2008

Version	Taille	Notes
XAMPP Windows 1.6.8 [kit de base]		Apache HTTPD 2.2.9, MySQL 5.0.67, PHP 5.2.6 + 4.4.9 + PEAR + Switch, Openssl 0.9.8i, PHPMyAdmin 2.11.9.2, XAMPP Control Panel 2.5, Webalizer 2.01-10, Mercury Mail Transport System v4.52, FileZilla FTP Server 0.9.27, SQLite 2.8.15, ADODB 4.98, Zend Optimizer 3.3.0, XAMPP Security, Ming. Pour Windows 2000, 2003, XP, VISTA. Voir aussi Lisez-moi.
Installateur	38 Mo	Installateur MD5 checksum: 9fe3b33e571ee57d02cb5ebdc888865f
ZIP	90 Mo	Archive ZIP MD5 checksum: 9509aacb8fedeb74684087a53276e5c5

2. Une fois le fichier téléchargé, exécutez-le pour lancer le programme d'installation.
3. Validez les différentes fenêtres, et les fichiers s'installent. À la fin, une fenêtre DOS s'affiche, afin de configurer les serveurs pour votre machine, et l'installation est terminée. Au moment de quitter l'installateur, celui-ci vous propose de lancer le panneau de contrôle de XAMPP, ce que vous pouvez faire (voir Figure 6.02).

Figure 6.02
Lancement du panneau de contrôle de XAMPP.



Une fois que tout est installé, il reste à configurer les serveurs et à apprendre à démarrer et stopper. Tout ceci s'accomplit depuis le panneau de contrôle.

1. Lancez le panneau de contrôle. Si l'installateur n'a pas placé d'icône sur votre Bureau, vous devez ouvrir le dossier où XAMPP a été installé (par défaut, C:\xampp) et double-cliquer sur l'icône xampp-control.exe (voir Figure 6.03).

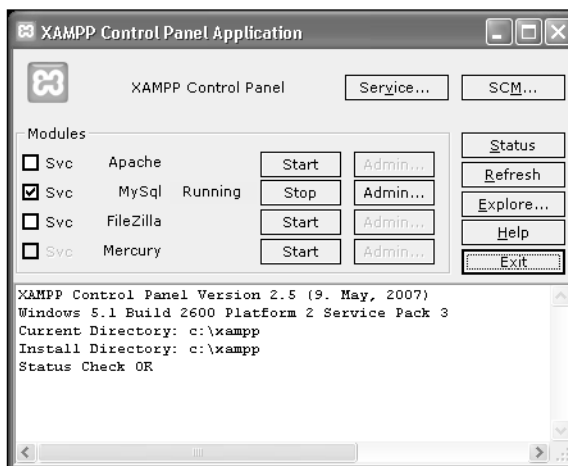


Figure 6.03
xampp-control.exe.

2. Dans ce panneau, chaque serveur dispose d'un bouton Start/Stop. Activez les serveurs Apache et MySQL s'ils ne le sont pas déjà. Il est fort probable que lors du premier démarrage du serveur Apache, le système de sécurité de Windows affiche une alerte vous demandant s'il faut bloquer le programme Apache HTTP Server. Ici cliquez sur le bouton Débloquer (voir Figure 6.04).

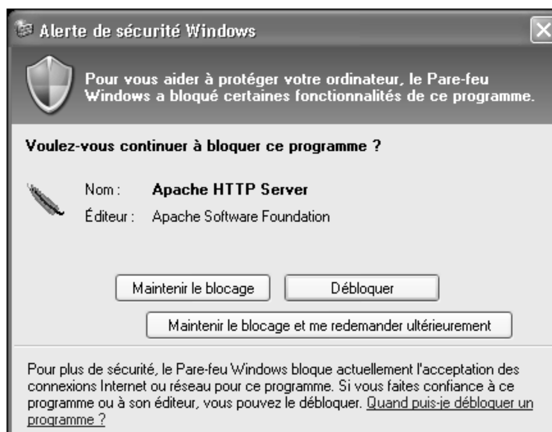
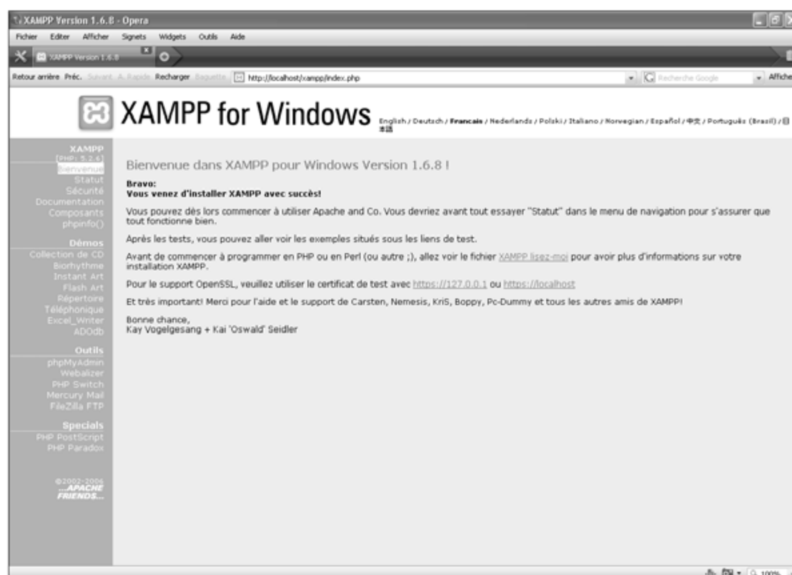


Figure 6.04
Déblocage de XAMPP.

3. Ouvrez votre navigateur Internet et saisissez l'adresse *http://localhost/*, ou cliquez simplement sur le bouton Admin du serveur Apache depuis le panneau de contrôle. L'adresse "localhost" est celle du serveur qui vient d'être installé sur votre ordinateur.

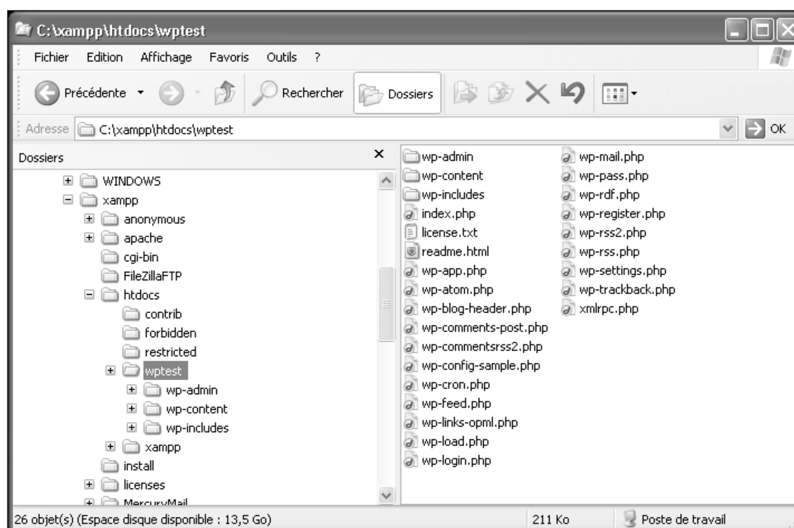
Sur cette page, vous devez choisir une langue. Une fois que c'est fait, le site s'ouvre avec l'interface d'administration de XAMPP (voir Figure 6.05).

Figure 6.05
Interface d'administration de XAMPP.



Dès lors, vous pouvez commencer à exploiter le serveur web, en plaçant les fichiers HTML, PHP et CSS dans un sous-dossier dédié à votre site, placé dans le C:\xampp\htdocs. Par exemple, pour installer un blog WordPress de test, vous pourriez placer les fichiers dans C:\xampp\htdocs\wptest (voir Figure 6.06).

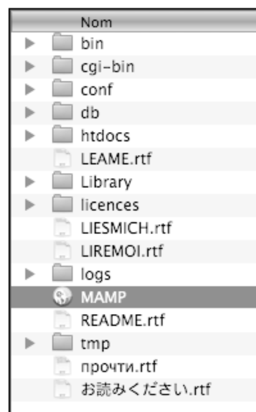
Figure 6.06
Dossier blog sur XAMPP.



Installer MAMP sur Mac OS X

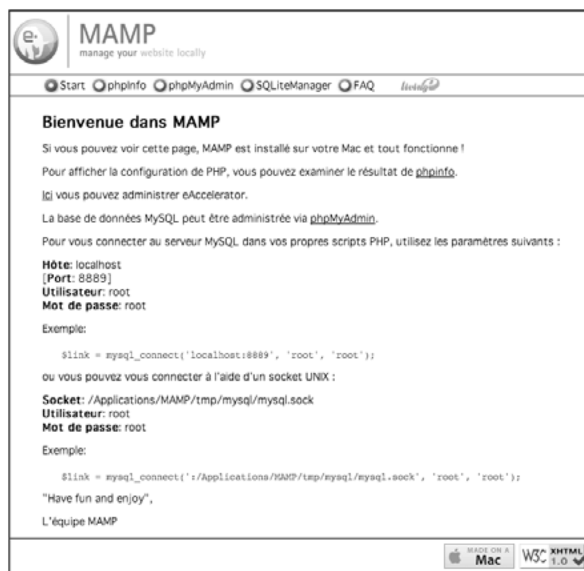
1. Téléchargez MAMP et installez-le dans votre dossier d'applications.
2. Cliquez sur l'icône de l'application MAMP pour démarrer automatiquement les serveurs Apache et MySQL et ouvrir par là même l'application (voir Figure 6.07).

Figure 6.07
Ouvrir MAMP.



3. Sur la fenêtre de l'application, cliquez sur le bouton Ouvrir la page d'accueil qui va ouvrir automatiquement une fenêtre dans votre navigateur. C'est celle de l'interface d'administration de MAMP (voir Figure 6.08).

Figure 6.08
Interface d'administration de MAMP.



Ensuite, le fonctionnement est le même que pour XAMPP. Vous placerez vos fichiers HTML, PHP et CSS dans un dossier rangé directement sous MAMP/htdocs.

Création de la base de données locale de WordPress

Avant d'installer WordPress, vous devez créer une base de données dans laquelle se placeront les tables qui contiendront les données du blog. Quel que soit le logiciel que vous utilisez, vous avez accès au phpMyAdmin *via* l'interface d'administration. Vous allez donc cliquer sur le lien vers phpMyAdmin, et de là créer votre base de données :

1. Trouvez la ligne Créer une base de données, en gras dans la section droite de l'interface. En dessous de cette ligne, dans le champ libre, écrivez "wordpress", qui sera le nom de la base de données de votre installation en local.

2. Cliquez sur le bouton Créer sans vous soucier du menu Interclassement. Votre base de test va alors être listée dans la colonne de gauche. Quittez phpMyAdmin : votre base de données est maintenant créée.

Installation de WordPress en local

Commencez par télécharger la dernière version de WordPress depuis WordPress Francophone (**www.wordpress-fr.net**) et placez-la directement dans le dossier htdocs de l'application (xampplite/htdocs ou MAMP/htdocs).

La procédure d'installation sera la même que sur un hébergement en ligne, si ce n'est que les données de connexion à la base de données seront adaptées à une installation locale. Celles-ci vous sont fournies sur l'interface d'administration de XAMPP ou de MAMP. Elles seront quasi identiques :

- Nom de la base : wordpress.
- Nom de l'utilisateur : root.
- Mot de passe : root pour MAMP ; ne saisissez rien pour XAMPP, l'espace doit rester vide.
- Nom de l'hôte : localhost.

Pour lancer le script d'installation, entrez l'adresse locale du fichier. Si vous avez choisi de placer les fichiers dans le /htdocs/wordpress, vous pourrez lancer WordPress en tapant *http://localhost/wordpress* dans votre navigateur. Vous serez alors directement redirigé vers la page d'installation de WordPress, où vous utiliserez la même procédure que pour un blog en ligne (voir Chapitre 2). Une fois installé, votre blog de test sera visible à cette même adresse.

Vous allez maintenant pouvoir faire tous les tests que vous voulez sur cette installation en local, sans avoir besoin d'être connecté à Internet, et sans que ce blog soit visible en ligne.

Alimenter votre base de données

Tout au long de cette section, vous allez tester continuellement votre thème. Il est donc important d'alimenter votre blog de test dès le départ avec quelques articles et une ou deux pages statiques.

Par défaut, WordPress installe un article et une page d'exemple, mais nous vous recommandons d'en créer plusieurs, avec des catégories, des images, des sous-pages, des tags… Bref, faites en sorte que ce blog soit plus vrai que nature, afin de ne pas être pris par surprise quand votre thème sera utilisé "pour de vrai".

Le contenu n'a aucune importance ici, vous pouvez donc utiliser des variantes du célèbre faux texte "Lorem Ipsum…". Celui-ci, par son manque de valeur sémantique combiné à sa grande ressemblance à un texte normal, est utilisé depuis longtemps pour construire des mises en page sans pour autant avoir le texte définitif sous la main. Vous trouverez une explication plus détaillée, ainsi qu'un très utile générateur automatique, sur le site **http://fr.lipsum.com/**.

Il existe sur Internet des extensions ou des fichiers qui vous permettent de remplir une base d'essai pour votre blog. C'est le cas notamment d'un fichier XML fourni par WPCandy (**http://wpcandy.com/articles/easier-theme-development-with-the-sample-post-collection.html**). Pour installer le fichier, il vous faut aller dans le menu Outils > Importer et choisir l'option WordPress, qui vous permet d'importer des articles, commentaires et autres éléments sur votre blog.

Une fois le fichier importé, retournez sur la page d'accueil de votre blog, vous allez découvrir toute une série de nouveaux articles, avec des catégories, commentaires, et j'en passe.

L'avantage de ce contenu "bidon" est qu'il va vous permettre de tester votre thème dans toutes les conditions possibles. Vous allez trouver des articles avec des listes, ordonnées et non ordonnées, des images calées à gauche, d'autres à droite. Bref, tout ce qu'il vous faut pour tester votre thème.

Création des fichiers du blog, les "fichiers de modèle"

À partir de maintenant, nous partons du principe que vous disposez d'un blog WordPress de test, qu'il soit en ligne ou en local. Nous allons pouvoir nous consacrer à la création du thème.

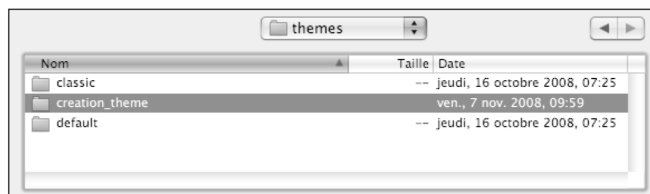
Nous avons vu au chapitre précédent comment était structuré un thème WordPress. Ici, nous allons passer directement à sa réalisation.

Création du dossier du thème

Vous allez commencer par créer le dossier qui va héberger les différents fichiers de votre blog. Pour cela, allez dans le dossier themes de votre installation (/htdocs/wordpress/wp-content/themes) et créez-y un nouveau dossier, que vous appellerez creation_theme (voir Figure 6.09).

Figure 6.09

Création du dossier de thème.



Création des premiers fichiers du thème

Vous allez ensuite créer les différents fichiers de "base" que va contenir votre thème. Nous avons vu précédemment que le fichier de base était index.php. Mais ici, nous allons séparer les différentes parties en une organisation logique et donc créer quatre fichiers : index.php, header.php, sidebar.php et style.css (la feuille de style du blog). Tous ces fichiers seront placés à la racine du dossier de votre thème, c'est-à-dire ici /htdocs/wordpress/wp-content/themes/creation_theme.

C'est le fichier index.php qui va référencer l'ensemble des données de la page web. Nous allons donc y renseigner différentes informations en premier, ce qui vous permettra de faire des tests en direct tout au long de la création du thème. En effet, si vous commencez par créer le fichier de l'en-tête, header.php, rien ne s'affichera sur le navigateur puisque le fichier index.php, tel que le présente la hiérarchie des modèles, n'existe pas – et, *in extenso*, le thème ne serait alors pas valide.

Notez que tous vos fichiers doivent être encodés en UTF-8, pour éviter les problèmes d'affichage sur les lettres accentuées et autres caractères spéciaux. La plupart des outils le font aujourd'hui par défaut, mais si votre éditeur est réglé pour enregistrer ses fichiers avec un encodage différent, par exemple ISO-8891-1, prenez soin de modifier ce réglage.

Création du modèle index.php

Nous allons commencer par le fichier PHP de base du thème : index.php. Le contenu de ce fichier va avoir la structure suivante :

- données de l'en-tête (importées depuis header.php) ;
- contenu du blog (récupéré par la boucle WordPress) ;
- colonne latérale (importée depuis sidebar.php) ;
- pied de page (importé depuis footer.php).

Création de la feuille de style CSS

Vous allez maintenant créer la feuille de style du blog. Il est très important de la créer dès le départ, sans quoi le thème n'apparaîtra pas dans le sélecteur de thème de l'interface d'administration de WordPress, et vous ne pourrez pas le choisir pour le tester.

Vous allez d'abord fournir différentes informations qui permettront à WordPress d'afficher votre thème parmi ceux qui sont disponibles. Pour cela, commencez le fichier avec les lignes de commentaire suivantes :

```
/*
Theme Name: Création Thème WordPress
Theme URI: http://www.fran6art.com/
Description: Thème créé de A à Z avec le livre Campus WordPress.
Version: 1.0
Author: Francis Chouquet
Author URI: http://www.fran6art.com/

Creation Theme by Francis Chouquet || http://www.fran6art.com
*/
```

Vous devez respecter à la lettre le nommage des différentes sections de l'en-tête de style. css, car c'est de là que WordPress va extraire les informations affichées dans son sélecteur de thème. Deux thèmes ne doivent pas présenter les mêmes informations, car cela provoque de la confusion dans le sélecteur.

Les sections obligatoires sont :

- `Theme Name` : le nom du thème ;
- `Theme URI` : l'adresse de la page web qui présente le thème ;
- `Description` : un descriptif du thème ;
- `Author` : le nom de l'auteur du thème ;
- `Author URI` : l'adresse du site de l'auteur.

Les sections optionnelles sont :

- `Version` : le chiffre de la version ;
- `Template` : le thème à utiliser (fonctionnalité avancée).

Enfin, vous pouvez terminer cet en-tête en ajoutant un texte libre, qui peut être une explication du fonctionnement, une licence d'utilisation, un avertissement… Ce texte ne sera pas repris par WordPress.

Notez bien que les informations présentées dans notre exemple concernent l'auteur de ces pages, mais vous pouvez et même devez les remplacer par les vôtres.

Ceci fait, enregistrez votre feuille de style et allez sur l'interface d'administration de votre blog, onglet Apparence et sous-menu Thèmes. Vous pouvez fermer le fichier, car nous ne toucherons pas aux CSS pour cette première partie, pour nous concentrer sur le code HTML et la sémantique des éléments affichés.

Votre thème est désormais disponible dans WordPress (voir Figure 6.10).

Figure 6.10
Votre thème apparaît dans WordPress.



Sélectionnez-le et activez-le. Pour le moment, vous n'avez qu'une page blanche.

Sur la page de l'interface d'administration de WordPress, chaque thème est associé à une image miniature censée le représenter. Si vous le souhaitez, vous pouvez d'ores et déjà créer cette miniature. Pour cela :

1. Créez une image aux dimensions 300 × 240 pixels, peu importe son contenu.
2. Appelez-la screenshot.png.
3. Placez-la au même niveau que les autres fichiers de votre thème.

Lorsque vous retournez sur la page d'administration des thèmes de WordPress, votre thème apparaît désormais accompagné de son image. Quand votre thème sera terminé, il sera toujours temps de créer une nouvelle image plus représentative, comme une capture d'écran du thème en action.

Le fichier index.php est créé, et votre blog de test est désormais configuré pour utiliser votre thème – même si, pour le moment, WordPress affiche une page blanche. Maintenant vous allez pouvoir ajouter du contenu à tous les fichiers déjà créés et ainsi découvrir toutes les informations qu'ils proposent.

Un conseil sur le nommage des règles CSS : durant la création de ce thème, nous utiliserons des termes anglais pour nommer les différentes règles CSS. En effet, il est souvent préférable de les avoir dans une langue connue par un maximum de personnes, au cas où vous souhaiteriez le proposer par la suite au téléchargement. Si vous employez une terminologie francophone, il sera très difficile par la suite de modifier le thème pour quelqu'un qui ne connaît pas le français.

Création du modèle header.php

Vous allez à présent créer le fichier header.php. C'est lui qui contient les informations de base qui vont permettre à la page web de bien s'afficher dans le navigateur. Cette partie comprend notamment :

- le DOCTYPE, qui fournit au navigateur les informations sur la spécification utilisée (XHTML, HTML 4, etc.) ;
- la balise HEAD et tout ce qu'elle contient : les balises META, le titre du blog, le lien vers la feuille de style et vers les flux RSS, etc. ;
- l'ouverture de la balise BODY, qui va englober l'ensemble du style et du contenu du blog.

Toutes ces informations constituent les bases pour créer votre thème. Sans le doctype, le navigateur ne saura pas comment afficher la page web et devra se débrouiller – au risque de mal interpréter vos balises. Sans la balise HEAD et ses différentes informations, le navigateur ne saura pas quel type de codage de caractère utiliser, comment s'appelle le blog, où sont les infos de style, où se trouve le flux RSS… Ces données sont importantes pour tout site Internet et se trouvent toujours en premier, dans l'en-tête de la page web.

Tout ceci implique que ces informations devront se retrouver sur toutes les pages du blog, et donc que le fichier header.php devra être inclus au début de chaque page.

Vous allez ouvrir votre éditeur de texte et créer le fichier header.php. N'oubliez pas de l'enregistrer directement dans le dossier du thème creation_theme. Vous allez y ajouter les lignes de code suivantes (vous trouverez toutes ces données sur **http://apprendre-wordpress.fr/livre**) :

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
  <head profile="http://gmpg.org/xfn/11">
```

La première ligne représente le doctype qui, comme son nom l'indique, renseigne sur le type du document à afficher, ce qui est essentiel au navigateur pour choisir avec quel moteur de rendu traiter la page : le mode de rendu respectueux des standards (Standard), ou le mode de compatibilité (Quirks), ce dernier se débrouillant comme il peut pour afficher la page au mieux.

Vous comprenez donc que le doctype est une part essentielle d'une page valide. Dans nos exemples, nous allons utiliser le mode Standard en combinaison avec le langage XHTML 1.0 Transitional. Ce dernier est une version plus permissive que le langage XHTML 1.0 Strict, en cela que certains éléments de HTML 4.01 interdits dans la version Strict sont acceptés en Transitional (<center>, <font> ou encore <strike>), ce qui permet aux développeurs habitués au HTML de faire une transition facile vers le XHTML, un langage plus propre et extensible.

Un mot sur l'importance de la validité d'une page. Avoir un site valide, que ce soit en XHTML ou en HTML, permet d'être sûr que celui-ci dispose d'un code propre et cohérent (mais pas forcément juste), et que son traitement sera facilité. Nous verrons comment valider notre thème une fois celui-ci terminé, en fin de chapitre.

La ligne suivant immédiatement le doctype ouvre la balise <html>, qui regroupe l'ensemble du document XHTML et doit être la racine de toute page XHTML valide. Par ailleurs, en XHTML, cette balise doit contenir un attribut xmlns, qui permet d'associer la balise `html` à l'espace de nom (namespace) de XHTML, à savoir **http://www.w3.org/1999/xhtml**.

La ligne suivante ouvre l'élément <head>, qui contient toutes les balises donnant des informations sur la page ou le site. Le projet WordPress étant un fervent soutien du format de mise en relation XFN, la plupart des thèmes intègrent le profil XFN, mais cela reste une option.

Nous allons donc remplir l'élément <head> avec les informations relatives au site. Ces informations étant pour la plupart tirées de la base de données de WordPress, cela va nous forcer à entreprendre notre exploration des marqueurs de modèle. Commençons par le titre du document, contenu dans la balise <title> :

```
<title>
  <?php bloginfo('name'); ?>
  <?php if ( is_404() ) : ?>
    &raquo; <?php _e('Not Found'); ?>
  <?php elseif ( is_home() ) : ?>
    &raquo; <?php bloginfo('description'); ?>
  <?php else : ?>
    <?php wp_title(); ?>
  <?php endif; ?>
</title>
```

Ici nous faisons en sorte que la balise `title` contienne quelques informations supplémentaires. Après le titre du blog, nous placerons un texte correspondant au contexte :

- dans le cas d'une page introuvable, le message "Not Found" ;
- dans le cas de la page d'accueil, le sous-titre/descriptif du blog ;
- dans les autres cas, un appel au marqueur `wp_title()`.

`wp_title()` est un marqueur qui, à son tour, renvoie du texte en fonction du contexte :

- dans le cas d'un article seul ou d'une page seule, son titre ;
- dans le cas d'une archive par date, la date en question ("2008", "2008 – Décembre", etc.) ;
- dans le cas d'une catégorie ou d'un label (tag), son nom ;
- dans le cas d'une page d'auteur, son identifiant public.

Ajoutez ensuite les lignes suivantes qui correspondent aux métadonnées du blog – c'est-à-dire des informations à propos des informations du blog. Toutes ces données vont être utilisées par le navigateur pour afficher correctement la page web. Nous y retrouvons notamment l'encodage de caractères à utiliser (`charset`), l'emplacement de la feuille de style à employer (`stylesheet`), ainsi que les différents formats pour utiliser les flux RSS :

```
<meta http-equiv="Content-Type" content="<?php bloginfo('html_type'); ?>;
charset=<?php bloginfo('charset'); ?>" />
<meta name="generator" content="WordPress <?php bloginfo('version'); ?>" />
<!-- leave this for stats -->
<link rel="stylesheet" href="<?php bloginfo('stylesheet_url'); ?>" type="text/
css" media="screen" />
<link rel="alternate" type="application/rss+xml" title="RSS 2.0" href="<?php
bloginfo('rss2_url'); ?>" />
```

```
<link rel="alternate" type="text/xml" title="RSS .92" href="<?php
bloginfo('rss_url'); ?>" />
<link rel="alternate" type="application/atom+xml" title="Atom 0.3"
href="<?php bloginfo('atom_url'); ?>" />
<link rel="pingback" href="<?php bloginfo('pingback_url'); ?>" />
```

Le marqueur WordPress employé ici est `php bloginfo` avec différentes variables :

- `html_type` : permet de connaître le type de HTML utilisé.
- `version` : permet de connaître la version de WordPress employée.
- `stylesheet_url` : donne l'URL de la feuille de style.
- `rss2_url` : fournit ici l'URL du flux RSS au format RSS 2.0.
- `rss_url` : fournit l'URL du flux RSS au format RSS 0.92.
- `atom_url` : fournit l'URL du flux RSS au format Atom 0.3.
- `pingback_url` : fournit l'URL des rétroliens.

Ensuite, renseignez l'emplacement et le format des archives utilisées pour votre blog :

```
<?php wp_get_archives('type=monthly&format=link'); ?>
```

Et, enfin, placez les quelques lignes suivantes :

```
<?php wp_head(); ?>
</head>
<body>
```

Le marqueur `wp_head()` que vous insérez ici est ce qu'on appelle un hook pour les extensions WordPress. C'est une sorte de repère qui sera utilisé par les développeurs d'extensions pour afficher convenablement les informations dans les différents fichiers du thème. Sans ce repère, l'extension ne pourra pas fonctionner dans l'en-tête.

En dessous, vous refermez la balise `head` et vous ouvrez la balise `body`, à l'intérieur de laquelle vous allez insérer l'ensemble du contenu visuel du blog. Pour ce qui est de l'en-tête, ce contenu va se limiter, comme souvent, au titre du blog et à sa description.

Vous enregistrez votre fichier que vous nommez header.php. Maintenant, vous allez devoir insérer le marqueur de modèle `get_header` dans le fichier index.php, puisque c'est lui qui va appelé les différents "blocs" ou "modules" du thème.

Vous allez donc ouvrir le fichier index.php et y insérer la ligne de code suivante:

```
<?php get_header(); ?>
```

Dans le fichier header.php, vous avez ouvert la balise `body`. Vous allez la refermer ici, dans le fichier index.php, et directement sous le marqueur qui appelle l'en-tête. Vous allez donc avoir la structure de code suivante :

```
    <?php get_header(); ?>
  </body>
</html>
```

Maintenant que nous avons rempli l’en-tête du fichier avec les informations nécessaires au navigateur, il est temps de mettre en place les balises qui afficheront le blog lui-même, petit à petit : titre, article, commentaire…

Insertion du titre du blog

Nous l’avons vu au chapitre précédent, la requête PHP qui va afficher le titre du blog est :

```
<?php bloginfo('name'); ?>
```

Vous allez donc insérer ce code directement sous l’ouverture de la balise `body` (voir Figure 6.11).

Figure 6.11
Insertion du marqueur du titre de blog.

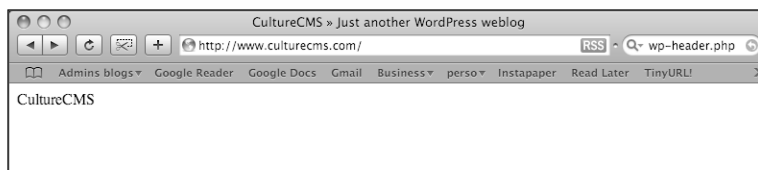
```
<?php wp_get_archives('type=monthly&format=link'); ?>

<?php wp_head(); ?>
</head>
<body>

<?php bloginfo('name'); ?>
```

Sauvegardez votre fichier et allez voir ce qui se passe dans votre navigateur. Le titre du blog apparaît en haut à gauche (voir Figure 6.12).

Figure 6.12
Affichage du titre du blog.

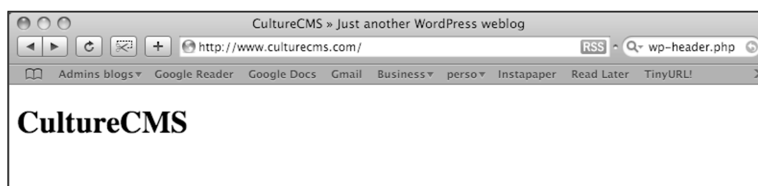


Vous allez maintenant donner au titre son premier style en lui attribuant une balise `h1`, qui est la balise la plus importante pour ce qui est des titres dans une page web. Pour ce faire, ajoutez `<h1>` devant la requête `php` et `</h1>` à la fin de cette même requête :

```
<h1><?php bloginfo('name'); ?></h1>
```

Sauvegardez votre fichier et rafraîchissez une nouvelle fois votre navigateur. Le titre du blog apparaît maintenant en plus gros, avec le style par défaut attribué aux titres `h1` (voir Figure 6.13).

Figure 6.13
Affichage du titre avec la balise *h1*.



À présent, vous allez mettre un lien sur ce titre qui va pointer sur la page d'accueil du blog. Cela se révélera très utile quand un visiteur qui sera sur une page intérieure du blog souhaitera revenir à la page d'accueil.

Pour faire pointer ce lien, il faut que vous retrouviez la structure classique d'un hyperlien, à savoir :

```
<a href="URL">texte</a>
```

Pour produire le "texte" cliquable, nous allons utiliser le marqueur dont nous venons de parler. Pour ce qui est de l'URL, nous nous servirons du même tag, `bloginfo()`, mais en lui appliquant le paramètre `url`, qui va dès lors afficher l'URL du blog plutôt que son nom. Entourez le tout par la balise `h1`, et vous obtenez la ligne de code suivante :

```
<h1><a href="<?php bloginfo('url'); ?>"><?php bloginfo('name'); ?></a></h1>
```

L'avantage d'utiliser des marqueurs de modèle plutôt que d'écrire directement le titre et l'URL est que vous pourrez employer votre thème sur n'importe quel blog, celui-ci affichera toujours les bonnes informations, car elles seront tirées directement depuis la base de données.

Une fois que vous avez modifié votre fichier header.php avec les nouvelles informations, enregistrez-le et rafraîchissez à nouveau la fenêtre de votre navigateur pour voir apparaître le lien sur le titre (voir Figure 6.14).

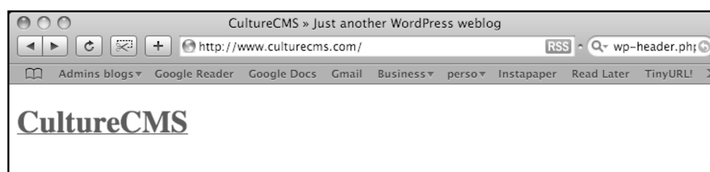


Figure 6.14
Affichage du titre du blog en lien.

Insertion de la description du blog

Vous allez maintenant ajouter la description du blog sous le titre. Ici, vous utiliserez une nouvelle fois le tag `bloginfo()` et vous lui attribuerez le paramètre `"description"`, tout simplement. Vous obtiendrez alors le marqueur de modèle complet suivant :

```
<?php bloginfo('description'); ?>
```

C'est donc ce morceau de code qui va afficher la description du blog. Insérez-le directement sous la ligne dédiée au titre du blog et sauvegardez votre fichier. Rafraîchissez votre navigateur : la description du blog apparaît sous son titre (voir Figure 6.15).

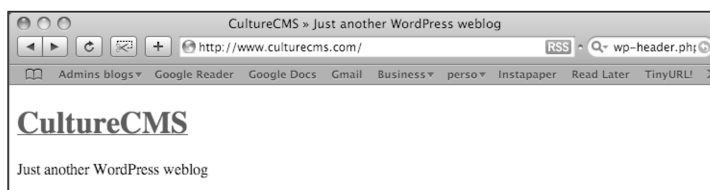


Figure 6.15
Affichage de la description du blog.

Mise en forme de l'en-tête du blog

Pour finir avec l'en-tête du blog, vous allez placer différentes balises CSS autour du titre et de la description du blog pour pouvoir leur donner un style spécifique par la suite.

Tout d'abord, vous placerez une balise `div` pour envelopper les deux lignes déjà créées. Elle permettra de donner un style complet pour l'en-tête. Pour cibler cette balise uniquement avec une règle CSS, nous ajoutons un attribut `"id"` ayant la valeur `"header"`. Cet attribut signifie que nous donnons un identifiant unique à cette balise et que seule cette balise aura cette valeur d'ID dans tout le code HTML. Vous obtenez alors la structure suivante :

```
<div id="header">
  <h1><a href="<?php bloginfo('url'); ?>"><?php bloginfo('name'); ?></a></h1>
  <?php bloginfo('description'); ?>
</div>
```

Sauvegardez votre fichier header.php.

Pour terminer, vous allez créer une nouvelle balise, qui va s'appeler `"page"`. Cette balise aura pour but de styler l'ensemble du contenu, là où la balise `"body"` met en forme l'ensemble de la page web.

```
<div id="page">
```

Schématiquement, d'un point de vue CSS, la structure doit être la suivante (voir Figure 6.16).



Figure 6.16
Structure CSS de la page web.

Vous allez donc ouvrir cette balise `page` directement sous la balise `body`. N'oubliez pas de refermer cette balise, dans le fichier index.php, directement au-dessus de la fermeture de la balise `body` :

```
    </div> <!-- fermeture page -->
  </body>
</html>
```

Pour ce qui est des informations d'en-tête, le code final sera donc le suivant :

```
<body>
  <div id="page">
    <div id="header">
      <h1><a href="<?php bloginfo('url'); ?>"><?php bloginfo('name'); ?></a></h1>
      <?php bloginfo('description'); ?>
    </div>
```

Nous en resterons là pour le moment avec le fichier header.php. En effet, le reste de la balise `body` (donc le reste de l'affichage du blog) sera pris en charge par le fichier index.php. Nous séparons donc physiquement le code en charge de l'affichage de la partie supérieure du design du blog et celui en charge du contenu lui-même.

Mise en place de la boucle WordPress

Vous allez maintenant insérer les différentes informations que contient la boucle WordPress. À partir d'ici, il est primordial que votre blog comprenne quelques articles pour vérifier que la boucle fonctionne correctement.

Ouvrez le fichier index.php que vous avez déjà créé et insérez les quelques lignes suivantes de la boucle WordPress sous le marqueur de modèle qui appelle le fichier header.php. En effet, dans notre exemple, le contenu va venir se positionner sous l'en-tête :

```
<?php get_header(); ?>

<div id="content">
  <?php if(have_posts()) : ?>
    <?php while(have_posts()) : the_post(); ?>
    <?php endwhile; ?>
  <?php else : ?>
    <h2 class="center">Aucun article trouvé. Essayez une autre recherche ?</h2>
    <?php include (TEMPLATEPATH . '/searchform.php'); ?>
  <?php endif; ?>
</div>
```

Comme nous l'avons vu au chapitre précédent, WordPress va vérifier s'il existe des articles à afficher. S'il y en a, il les affichera, sinon il vous proposera de faire une recherche. Ici, nous en avons également profité pour ajouter une balise `"content"` à la boucle WordPress. Cela permettra par la suite une meilleure personnalisation de l'affichage du contenu du blog.

Mais, dans ces quelques lignes, vous n'avez que les conditions de chargement des articles, pas l'affichage de ceux-ci. Vous devez maintenant placer aux bons endroits les marqueurs de modèle qui vont afficher le contenu même de ces articles.

Affichage du titre des articles

Les deux premières lignes de la boucle WordPress vérifient s'il y a des articles à afficher :

```
<?php if (have_posts()) : while (have_posts()) : the_post(); ?>
```

`have_posts()` et `the_post()` sont deux marqueurs de WordPress qui ont une importance primordiale pour la boucle. Sans eux, rien ne s'affiche.

Comme tous les marqueurs de modèle, ce sont là de simples raccourcis de fonctions PHP internes à WordPress qui simplifient grandement la récupération des données :

- `have_posts()` : renvoie la valeur VRAI ou FAUX selon qu'il reste ou non des articles à afficher selon le contexte en cours.
- `the_post()` : récupère l'identifiant interne de l'article suivant dans la liste des articles à afficher, ainsi que ses données (titre, contenu, permalien).

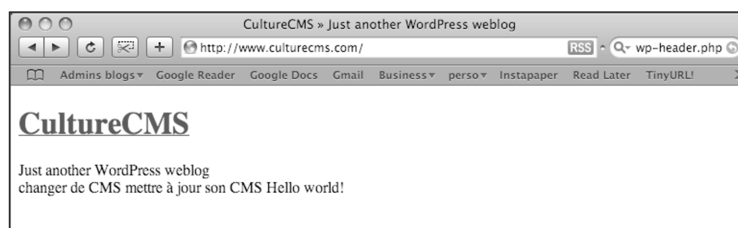
De fait, dès que le marqueur est appelé, toutes les données liées à l'article sont chargées et prêtes à être utilisées par les autres marqueurs dédiés. La boucle `while()` s'assure que tous les articles nécessaires passent par ce processus ; il est donc temps de mettre en place les marqueurs qui extraient le contenu de l'article en cours.

Le marqueur de modèle qui sera utilisé en premier est assez explicite :

```
<?php the_title(); ?>
```

Insérez-le donc directement sous les deux lignes précédemment citées, dans la boucle. Enregistrez les modifications et affichez la page web de votre blog pour voir le résultat. Maintenant, les différents titres de vos articles apparaissent les uns après les autres (voir Figure 6.17).

Figure 6.17
Affichage des titres des articles.



Comme vous l'avez fait pour le titre du blog, vous allez convertir ces titres d'articles en liens, afin de permettre au visiteur de cliquer sur le titre de l'article pour se rendre sur sa page – et ses commentaires. Vous allez modifier la ligne que vous venez d'insérer pour utiliser celle-ci :

```
<h2><a href="<?php the_permalink(); ?>" title="<?php the_title(); ?>"><?php
the_title(); ?></a></h2>
```

Vous avez ici ajouté le lien vers le permalien de l'article *via* le marqueur de modèle `the_permalink()`, et vous avez affiché le titre de l'article grâce au marqueur de modèle `the_title()`. Enfin, vous avez attribué une balise de titre qui sera ici H2 pour le titre affiché. La balise `h1` est réservée au titre du blog, et sur l'échelle d'importance des titres, on estime que ce sont ceux des articles qui doivent primer. On leur attribue donc un niveau H2.

Le résultat donne alors une liste de titres d'articles qui sont maintenant cliquables (voir Figure 6.18).

Figure 6.18

Affichage des titres des articles avec lien.



Vous pouvez même aller sur l'URL de l'article en question en cliquant sur le lien dès maintenant. Vous remarquez que l'affichage est identique à celui de la page d'accueil, si ce n'est qu'il n'y a que le titre de l'article sur lequel vous avez cliqué qui apparaît. C'est tout à fait normal, puisque si vous reprenez la hiérarchie des modèles de WordPress, quand il n'y a pas de fichier single.php créé, WordPress va utiliser le fichier index.php.

Affichage du contenu des articles

Pour afficher le contenu des articles, vous avez le choix entre deux marqueurs de modèle :

- `<?php the_content(); ?>` va afficher tout le contenu de l'article (jusqu'à la balise MORE).
- `<?php the_excerpt(); ?>` va afficher uniquement les 55 premiers mots de l'article (sans prendre MORE en compte).

Les résultats à l'affichage sont donc différents (voir Figures 6.19 et 6.20).

Pour la création de ce thème, vous allez utiliser `the_content()`. Vous l'insérerez directement sous la ligne de code pour le titre des articles à afficher :

```
<h2><a href="<?php the_permalink(); ?>" title="<?php the_title(); ?>">
<?php the_title(); ?></a></h2>

<?php the_content(); ?>
```

Sauvegardez votre fichier, et rafraîchissez le navigateur ; le contenu de vos articles est désormais visible.

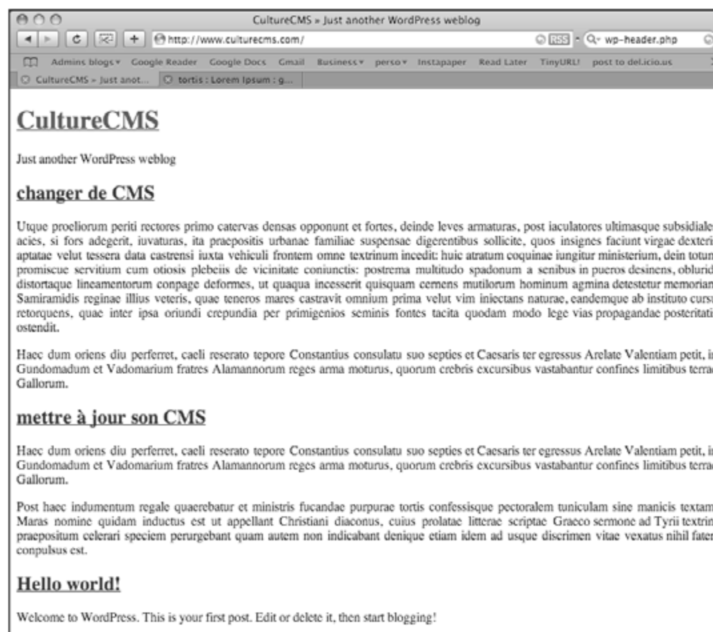


Figure 6.19

Affichage de la page d'accueil avec l'utilisation du tag *the_content()*.

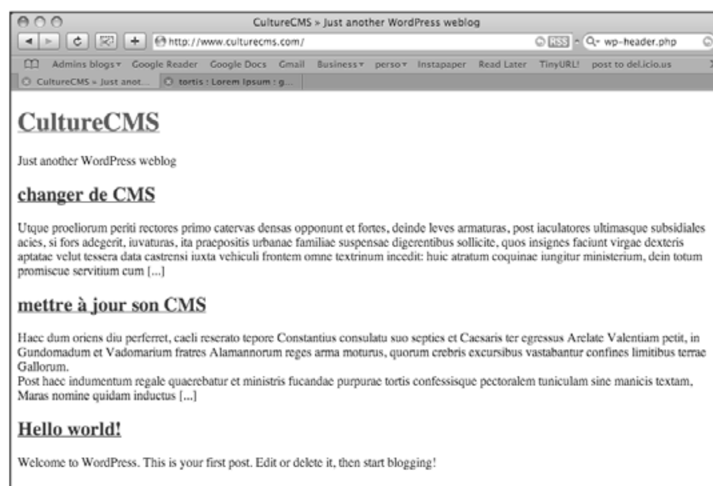


Figure 6.20

Affichage de la page d'accueil avec l'utilisation du tag *the_excerpt()*.

Mise en forme du contenu

Vous allez maintenant "baliser" les différents éléments insérés, pour pouvoir les personnaliser au mieux au moment de la création du style du blog.

Le titre des articles est déjà balisé avec le titre H2. Par contre, vous devrez mettre en place une balise pour le contenu même, mais aussi pour envelopper l'ensemble des articles, un

par un. Vous allez donc ajouter une balise `post` autour des éléments insérés pour un article, c'est-à-dire le titre et le contenu :

```
<div class="post">
  <h2><a href="<?php the_permalink(); ?>" title="<?php the_title(); ?>">
  <?php the_title(); ?></a></h2>
  <?php the_content(); ?>
</div>
```

Ici, nous allons utiliser une classe plutôt qu'un ID. Cela pour une raison simple : un ID est un identifiant unique, qui ne sera utilisé qu'une seule fois, alors qu'une classe peut être réutilisée autant de fois qu'on veut. Comme nous allons mettre en forme l'affichage des articles, il est important d'utiliser une classe, étant donné que nous prévoyons d'écrire plus d'un article.

Cette balise `post`, vous allez l'associer à un marqueur de modèle qui vous permettra de récupérer l'identifiant de l'article affiché. Ce marqueur de modèle est `the_ID()`, et vous allez l'insérer directement à l'intérieur d'une balise ID `post` :

```
<div class="post" id="post-<?php the_ID(); ?>">
```

Ici, vous séparez le style pour l'ensemble des articles, individuellement, avec la classe `post`, mais vous êtes également capable de personnaliser certains articles, en particulier grâce à l'ID "post-identifiant de l'article". L'ensemble va donc ressembler au code qui suit :

```
<div class="post" id="post-<?php the_ID(); ?>">
  <h2><a href="<?php the_permalink(); ?>" title="<?php the_title(); ?>">
  <?php the_title(); ?></a></h2>
  <?php the_content(); ?>
</div>
```

Ceci fait, vous allez ajouter une balise autour du contenu même, balise que vous appellerez `post_content` :

```
<div class="post_content">
  <?php the_content(); ?>
</div>
```

Vous employez ici des classes CSS, car les informations qui leur sont associées seront réutilisées pour tous les articles du blog.

À ce stade de la création du thème, le code global de la boucle WordPress est le suivant :

```
<div id="content">
  <?php if(have_posts()) : ?><?php while(have_posts()) : the_post(); ?>
  <div class="post" id="post-<?php the_ID(); ?>">
    <h2><a href="<?php the_permalink(); ?>" title="<?php the_title(); ?>">
    <?php the_title(); ?></a></h2>
    <div class="post_content">
      <?php the_content(); ?>
    </div>
  </div>
  <?php endwhile; ?>
  <?php else : ?>
```

```
  <h2 class="center">Introuvable</h2>
  <p class="center">D&eacute;sol&eacute;, mais vous cherchez quelque chose qui
ne se trouve pas ici .</p>
  <div id="searchno"> <?php get_search_form(); ?></div>
  <?php endif; ?>
</div>
```

Notez qu'en PHP la syntaxe ":" permet de faire un raccourci en évitant d'utiliser les accolades qui encadrent le contenu d'une fonction, mais obligeant du coup à fermer explicitement les mots-clefs idoines (`endif` pour `if`, `endwhile` pour `while`, etc.).

Ainsi, en écrivant le début de la boucle sans ce raccourci syntaxique, cela donnerait :

```
if (have_posts()) {
  while (have_posts()) {
    the_post();
    the_content();
    }
  else {
    echo "Introuvable";
  }
```

La décision de baser la boucle sur cette syntaxe résulte de l'envie de compartimenter autant que possible le code PHP et faire en sorte qu'il ressemble au minimum à du code PHP, qui effraie facilement les débutants, et plus à du code HTML.

Affichage des métadonnées des articles

Il vous reste maintenant une seule chose à intégrer pour que l'affichage du contenu soit complet : les métadonnées des articles. Ces données comptent entre autres les catégories, la date de publication, l'auteur de l'article, ainsi que le nombre de commentaires pour chaque article.

Les marqueurs de modèle utilisés pour afficher ces informations sont :

- date : `the_time()` ;
- auteur : `the_author()` ;
- catégories : `the_category()` ;
- commentaires : `comments_popup_link`.

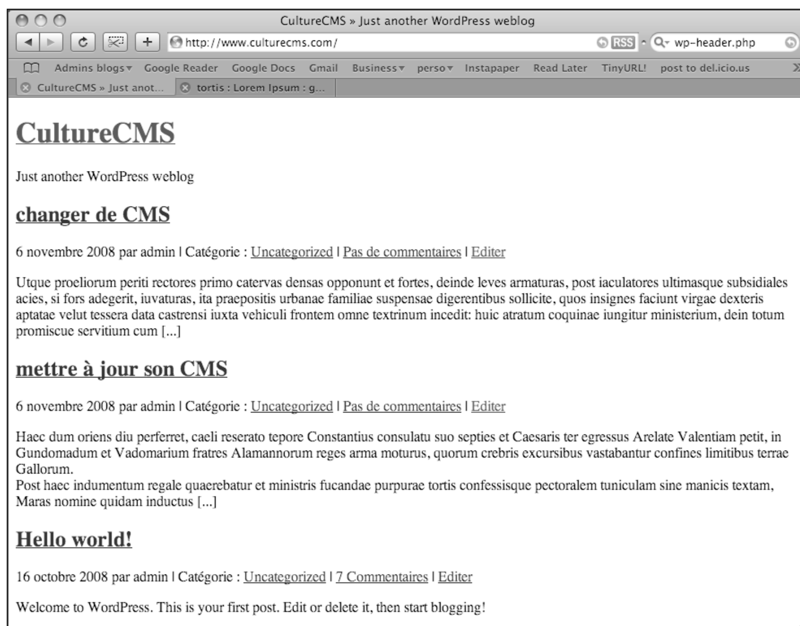
Vous allez placer ces différentes informations sous le titre de l'article, de manière qu'elles s'affichent de façon cohérente et lisible. Le code employé pour afficher ces données sera le suivant :

```
<p class="postmetadata">
  <?php the_time('j F Y') ?> par <?php the_author(); ?> | Cat&eacute;gorie :
<?php the_category(', '); ?> | <?php comments_popup_link('Pas de commentaires',
'1 Commentaire', '% Commentaires'); ?> <?php edit_post_link('Editer', ' &#124;
', ''); ?>
</p>
```

Enregistrez les modifications et vérifiez les changements dans votre navigateur. Votre blog ressemble désormais à ceci (voir Figure 6.21).

Figure 6.21

Affichage des métadonnées des articles.



Tout d'abord, vous avez créé une enveloppe pour ces informations. Nous l'avons appelée "postmetadata". Nous utilisons ici le tag P et non plus DIV pour des raisons de sémantique. La balise `P` correspond à un paragraphe. Cette balise est refermée à la fin de l'affichage de ces données.

Ensuite, le marqueur `the_time` va permettre l'affichage de la date. Ici, différents paramètres sont définis pour choisir le format de cette date. Vous trouverez tous les formats disponibles à l'adresse http://php.net/date. Puis vient `the_author()`, pour l'auteur de l'article.

Arrive enfin l'affichage du nombre de commentaires. C'est le marqueur `comments_popup_link()` qui est utilisé. Nous lui avons attribué trois paramètres différents pour trois cas :

- "pas de commentaires" s'affichera dans le cas où il n'y a pas de commentaires pour l'article.
- "1 commentaire" dans le cas où il n'y a qu'un seul commentaire.
- "% commentaires" pour tous les articles qui auraient plus d'un commentaire, % étant remplacé par le nombre de commentaires effectifs.

Cette personnalisation permet d'avoir un message différent selon le contexte.

Sous ces métadonnées, vous avez enfin ajouté un lien vers l'édition de chacun des articles. Cela peut se révéler très utile pour aller modifier un article directement à partir de la page d'accueil du blog par exemple (à condition d'être connecté). Le marqueur employé ici est `edit_post_link()`.

Insertion des mots-clefs

Si vous utilisez les labels (tags) sur votre blog, vous pouvez également les afficher ici. Pour ne pas les mélanger avec les métadonnées, vous allez intégrer cette information avec le marqueur de modèle `the_tags()` :

```
<p class="tags"><?php the_tags(); ?></p>
```

Ce code sera inséré après la fermeture de la classe `post_content` et avant la fermeture de la classe `post` :

```
  <div class="post_content">
    <?php the_content(); ?>
  </div>
  <p class="tags"><?php the_tags(); ?></p>
</div>
```

Vous avez balisé les mots-clefs avec une classe `tags`, et le marqueur utilisé ne comprend aucun paramètre particulier. Enregistrez les modifications et rafraîchissez votre navigateur pour voir le résultat (voir Figure 6.22).

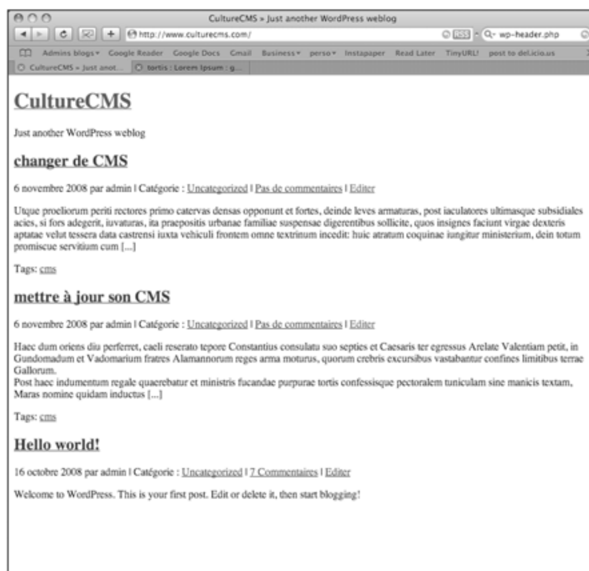


Figure 6.22
Visualisation des mots-clefs au niveau de l'article.

Création de la colonne latérale du blog

La colonne latérale, ou sidebar en anglais, va regrouper toute information qui pourrait être utile au visiteur. Pour l'heure, vous allez créer une colonne latérale simple, qui va rassembler l'ensemble des données utiles ou importantes que vous pouvez afficher. Pour votre propre blog, ce sera à vous de choisir les éléments que vous estimez utiles ou importants.

Commencez par ouvrir le fichier sidebar.php que vous avez créé au début de ce chapitre. Pour le moment, il n'a aucun contenu. Vous allez donc créer une balise avec une classe qui va envelopper cette colonne latérale. Vous l'appellerez `sidebar`. Nous avons choisi une classe ici pour pouvoir réutiliser son style sur d'autres pages du blog. À cette balise vous ajouterez une liste non ordonnée. Cela vous permettra de prendre l'ensemble du contenu de la colonne latérale pour une liste globale. En effet, sémantiquement, on peut considérer l'ensemble du contenu de la colonne comme une liste d'informations. Chaque bloc d'informations sera un élément de cette liste.

Le code à insérer est le suivant :

```
<div class="sidebar">
  <ul>
  </ul>
</div>
```

Tout ce qui sera ajouté par la suite devra être compris entre ces deux balises UL.

Vous allez également devoir insérer le marqueur de modèle pour la colonne latérale dans le fichier index.php. Directement au-dessus de la fermeture de la balise `page`, insérez le code suivant :

```
<?php get_sidebar(); ?>
```

Vous devriez avoir la structure suivante :

```
<?php endif; ?>
</div>
<?php get_sidebar(); ?>
  </div> <!-- fermeture page -->
  </body>
</html>
```

Insertion du formulaire de recherche

Vous allez à présent remplir votre colonne avec un formulaire de recherche. Il est toujours important de bien positionner ce formulaire pour permettre au visiteur d'approfondir sa recherche quand il ne trouve pas une information.

Avant d'insérer quoi que ce soit dans le fichier sidebar.php, vous devez créer un nouveau fichier qui correspondra à ce formulaire de recherche. Nous considérons ce formulaire comme un élément à part entière du blog, qui pourra être réutilisé sur d'autres pages, la page 404, par exemple ; par conséquent, il est plus facile de créer un fichier distinct et de l'ajouter grâce à une requête plutôt que de copier l'ensemble du code chaque fois.

Ouvrez donc un nouveau fichier et enregistrez-le au même niveau que les autres fichiers du blog, en le nommant searchform.php. Dans ce fichier, insérez les lignes de code suivantes :

```
<form method="get" id="searchform" action="<?php bloginfo('home'); ?>/">
  <div>
    <input type="text" value="<?php the_search_query(); ?>" name="s" id="s" />
    <input type="submit" id="searchsubmit" value="Chercher" />
  </div>
</form>
```

Vous venez de créer un formulaire (balise form). Vous y avez placé deux balises input qui forment le formulaire de recherche en tant que tel : le champ de recherche (type text) qui va utiliser le marqueur de modèle the_search_query(), et le bouton (type submit) pour effectuer la recherche, appelé Chercher.

Votre fichier searchform.php est désormais créé. Vous pouvez le fermer, vous n'aurez plus à y toucher.

Maintenant, vous allez le relier à la colonne latérale pour qu'il puisse s'afficher sur le blog.

Le marqueur que nous allons utiliser ici est :

```
<?php get_search_form(); ?>
```

Vous entourerez ensuite ce code d'une balise li. En effet, vous êtes dans une liste que vous avez générée un peu plus tôt (les balises UL), vous devez donc créer une ligne ici. Vous en profiterez également pour lui donner un attribut "id" que vous appellerez search. Le code final donne le résultat suivant :

<li id="search"><?php get_search_form(); ?>
Insérez ce code directement sous l'ouverture de la liste :

```
<div class="sidebar">
   <ul>
         <li id="search"><?php get_search_form(); ?></li>
   </ul>
</div>
```

Enregistrez votre fichier sidebar.php et ouvrez votre navigateur pour voir le résultat. Le formulaire de recherche, premier élément de la colonne, apparaît sous le dernier article ; c'est tout à fait logique, car tant que nous n'avons pas stylé les éléments du blog avec CSS, la page les affiche selon l'ordre dans lequel les éléments HTML ont été écrits. Le code de la colonne a été placé après celui des articles, donc le premier élément de celle-ci apparaît sous le dernier article affiché (voir Figure 6.23).



Figure 6.23
Affichage du formulaire de contact.

Insertion du calendrier

Le marqueur de modèle utilisé pour afficher le calendrier de WordPress est get_calendar(). Vous allez l'insérer sous le formulaire de recherche, toujours accompagné d'une balise de ligne et d'un titre h2 :

```
<li id="calendar"><h2>Calendrier</h2>
  <?php get_calendar(); ?>
</li>
```

Enregistrez les modifications, et rafraîchissez la fenêtre de votre navigateur. Le calendrier apparaît maintenant sous le formulaire de recherche (voir Figure 6.24).

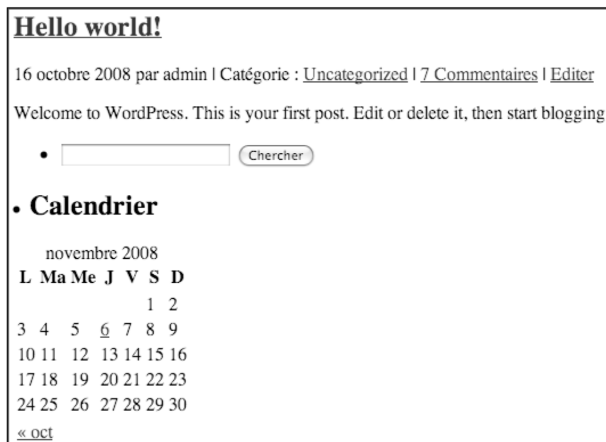


Figure 6.24
Affichage du calendrier.

Insertion des catégories

Le marqueur de modèle WordPress utilisé pour afficher la liste des catégories est `wp_list_categories()`. Vous allez l'insérer de la manière suivante :

```
<?php wp_list_categories('sort_column=name&optioncount=1&hierarchical=0&title_
li=<h2>Catégories</h2>'); ?>
```

Le marqueur de modèle est associé ici à plusieurs paramètres distincts pour avoir une certaine personnalisation de cette liste :

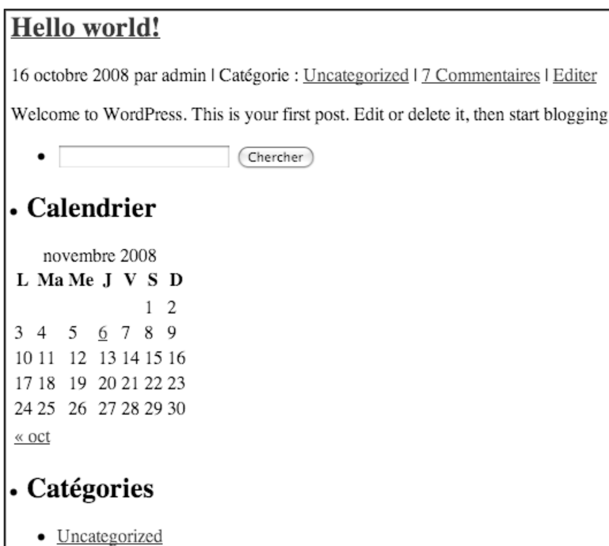
- `sort_column=name` : trie la liste par ordre alphabétique.
- `optioncount=1` : affiche le nombre de billets pour chaque catégorie. Si vous aviez mis le chiffre 0 à la place du 1, le nombre de billets ne s'afficherait pas.
- `hierarchical=0` : interdit l'affichage des sous-catégories. Si vous voulez les voir apparaître, mettez 1 à la place de 0.
- `title_li=xxx` : précise comment afficher le titre associé au tag. Ici, nous utiliserons de préférence un titre H2 que nous afficherons, mais chacun remplit cet espace comme il le souhaite.

Ici, il n'y a pas besoin de balise de ligne "li", cette balise étant déjà intégrée dans le marqueur de modèle.

Une fois le code inséré sous le calendrier, enregistrez les modifications et rafraîchissez la fenêtre de votre navigateur. Les catégories apparaissent désormais sous le calendrier (voir Figure 6.25).

Figure 6.25

Affichage des catégories.



Insertion des pages du blog

L'insertion de la liste des pages statiques sert souvent pour créer un menu de navigation du blog. Ici, vous allez ajouter les différentes pages du blog en une simple liste, grâce au marqueur de modèle `wp_list_pages()`.

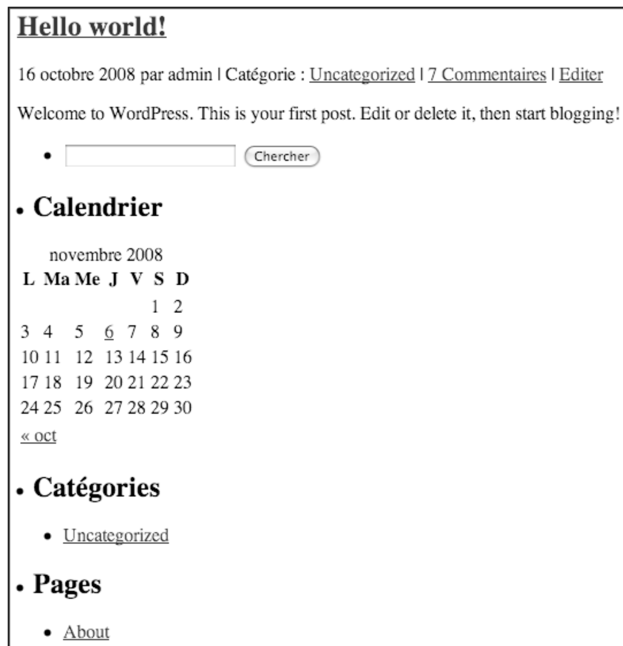
Ce sera exactement le même fonctionnement que pour les catégories. Le marqueur de modèle offre déjà la structure des lignes de la liste, et vous allez modifier le titre dans les paramètres pour avoir un titre dans une balise `h2`. Le code est le suivant :

```
<?php wp_list_pages('title_li=<h2>Pages</h2>'); ?>
```

Avec la version 2.7 de WordPress, une fonction très ressemblante a fait son apparition ; il s'agit de `wp_page_menu()`. Ce marqueur de modèle est plus complet que `wp_list_pages()` car il permet d'afficher, en plus des pages du blog, un lien vers la page d'accueil et de mettre en "relief" la page active.

Enregistrez le tout et rafraîchissez une nouvelle fois votre navigateur. La liste des pages du blog apparaît maintenant sous les catégories (voir Figure 6.26).

Figure 6.26
Affichage des pages du blog.



Insertion des archives

Le marqueur de modèle utilisé pour l'insertion des archives est `wp_get_archives()`. Vous allez ajouter un paramètre à ce modèle pour afficher ces archives par mois. Vous obtenez alors le code suivant :

```
<?php wp_get_archives('type=monthly'); ?>
```

Pour afficher les archives correctement, vous devez entourer ce marqueur par une balise de liste `ul`. Les mois s'afficheront alors sous forme de liste – une fois encore pour des questions de sémantique. Vous devez préciser ici la racine de la liste, car le marqueur de modèle ne la met pas en place de manière automatique. Enfin, vous allez insérer un titre au début et envelopper le tout par une balise de liste. Le résultat est le suivant :

```
<li><h2>Archives</h2>
  <ul>
    <?php wp_get_archives('type=monthly'); ?>
  </ul>
</li>
```

Insérez ce code sous la liste des pages du blog et regardez le résultat sur votre navigateur. La liste complète des archives rangées par mois apparaît alors en bas de la page (voir Figure 6.27).



Figure 6.27

Affichage des archives du blog.

Insertion de la blogoliste

Pour afficher la blogoliste, vous allez utiliser le marqueur de modèle `get_links_list()`. Cette fonction gère automatiquement le titre et la liste, ce n'est donc pas la peine d'ajouter quoi que ce soit ici. Vous insérerez le code tel quel directement sous les archives :

```
<?php get_links_list(); ?>
```

Enregistrez votre fichier et rafraîchissez votre navigateur. La blogoliste apparaît maintenant sous les archives (voir Figure 6.28).



Figure 6.28

Affichage de la blogoliste.

Insertion des informations de connexion

Sur certains blogs, il est possible de s'enregistrer pour laisser par exemple des commentaires. Il est donc important de laisser un lien dans la colonne latérale. Le marqueur de modèle utilisé pour l'enregistrement est `wp_register()`.

Il peut être utile également d'avoir un lien de connexion sur chacune des pages de votre blog. C'est un très bon raccourci pour se connecter à l'interface d'administration. Pour ce faire, vous devrez utiliser le marqueur de modèle `wp_loginout()` qui affichera un lien Connexion quand vous ne serez pas connecté, et un lien Déconnexion quand vous serez connecté.

Vous devrez aussi ajouter à la fin de la colonne latérale le crochet (hook) `wp_meta()` pour les extensions, qui leur permettra d'afficher leur contenu dans la colonne. C'est le même raisonnement que pour `wp_head()`, vu dans l'en-tête.

Le code à insérer sous la blogoliste est le suivant :

```
<li><h2>Infos Meta</h2>
  <ul>
    <?php wp_register(); ?>
    <li><?php wp_loginout(); ?></li>
    <?php wp_meta(); ?>
  </ul>
</li>
```

Vous commencez par créer une ligne `li` de la liste qui va englober le tout. Vous ajoutez un titre H2, et vous créez ensuite une liste `ul` pour pouvoir afficher les différentes informations méta. `wp_register()` gère automatiquement la ligne, alors que vous devez en ajouter une au modèle `wp_loginout()`.

Une fois votre code inséré, vous pouvez enregistrer votre colonne latérale et voir s'afficher les métadonnées dans votre navigateur. Ce que vous verrez dépendra du fait que vous soyez connecté ou non (voir Figure 6.29).



Figure 6.29
Affichage des données de connexion.

Insertion des flux RSS

Les dernières informations que vous allez insérer dans votre colonne sont les liens vers les flux RSS pour les articles et les commentaires. Vous les avez vus précédemment dans le thème default, mais ils étaient situés dans le pied de page. Ici, nous allons les inclure dans la colonne latérale, afin de leur donner une meilleure visibilité que dans le pied de page. Le marqueur de modèle utilisé sera encore `bloginfo()`, mais avec des paramètres différents :

- `rss2_url` pour le flux RSS des articles ;
- `comments_rss2_url` pour le flux RSS des commentaires des articles.

Ajoutez le code suivant :

```
<li><h2>Abonnez-vous au blog !</h2>
  <ul>
    <li><a href="<?php bloginfo('rss2_url'); ?>" title="Flux RSS des
articles">Flux RSS des articles</a></li>
    <li><a href="<?php bloginfo('comments_rss2_url'); ?>" title="Flux RSS des
commentaires">Flux RSS des commentaires</a></li>
  </ul>
</li>
```

Nous reprenons le même schéma que précédemment. Vous créez une ligne de liste et vous lui donnez un titre, ici Abonnez-vous au blog !. Ensuite, vous ouvrez une nouvelle liste et vous insérez un lien qui pointe vers les deux flux en ajoutant les marqueurs de modèle pour avoir les URL des flux RSS.

Enregistrez votre fichier et rafraîchissez votre navigateur. Les liens vers les flux RSS apparaissent maintenant sous les informations méta (voir Figure 6.30).

Figure 6.30
Affichage des flux RSS.



Widgetisation de la colonne latérale

Première méthode

Les widgets, que vous avez pu découvrir dans la partie "utilisateur", sont apparus avec la version 2.2 de WordPress. Ils s'installent dans votre colonne latérale directement depuis l'interface d'administration de WordPress et viennent y remplacer le contenu que vous avez mis dans votre colonne latérale. Mais, pour pouvoir installer ces widgets, il faut que votre colonne latérale soit "widgetisable". Pour cela, vous devez insérer quelques morceaux de code dans différents fichiers.

Tout d'abord, vous allez ajouter un peu de code dans votre fichier sidebar.php pour que WordPress sache que vous souhaitez utiliser les widgets dans votre colonne latérale. Insérez donc le code suivant directement sous la balise `ul`, qui ouvre la liste non ordonnée de la colonne latérale :

```
<div class="sidebar">
  <ul>
    <?php if ( !function_exists('dynamic_sidebar')
        || !dynamic_sidebar() ) : ?>
```

Et le code suivant, juste avant la fermeture de la balise `ul` de la liste de la colonne latérale :

```
<?php endif; ?>
```

Ainsi, c'est toute la colonne latérale qui sera widgetisée. Si vous utilisez un widget, il apparaîtra dans votre colonne latérale, sinon c'est le contenu du fichier sidebar.php qui s'affichera.

Mais avant de pouvoir afficher des widgets, vous devez "activer" ce changement, et celui-ci sera actif *via* un fichier appelé functions.php. Vous allez donc créer un fichier functions.php et dans ce fichier, vous allez insérer le code suivant :

```
<?php
if ( function_exists('register_sidebar') )
  register_sidebar();
?>
```

Ce code va dire à WordPress d'afficher la colonne latérale dynamique, donc les widgets. Cette fonction est valable dans le cas où vous n'avez qu'une seule zone à widgetiser, ce que nous comptons faire pour notre thème. Cependant, s'il y a plusieurs zones de votre blog dans lesquelles vous souhaitez avoir des widgets, vous utiliserez le code suivant :

```
<?php
if ( function_exists('register_sidebars') )
  register_sidebars(4);
?>
```

Vous remarquerez que les seules différences entre ces deux codes sont le "s" à la fin de `sidebar` et l'ajout du chiffre 4 en argument. Ce chiffre équivaut au nombre de zones widgetisées que vous mettrez en place. Si vous n'en voulez que deux, par exemple, vous choisirez le chiffre 2.

Si vous avez plusieurs zones widgetisées, vous devrez aussi modifier le contenu du fichier sidebar.php ou d'autres fichiers pour y positionner les différentes zones. Vous reprenez alors le même format que celui qui était présenté précédemment, sauf que vous y ajoutez le chiffre de la zone à widgetiser derrière `dynamic_sidebar`. Par exemple, si vous souhaitez créer

trois zones widgetisées, vous allez insérer les trois morceaux de code suivants aux endroits voulus, sans oublier `php endif` pour refermer la zone à widgetiser :

```
<?php if ( function_exists('dynamic_sidebar') && dynamic_sidebar(1) ) : ?>
<?php if ( function_exists('dynamic_sidebar') && dynamic_sidebar(2) ) : ?>
<?php if ( function_exists('dynamic_sidebar') && dynamic_sidebar(3) ) : ?>
```

Vous allez maintenant vérifier que tout fonctionne bien. Allez dans l'interface d'administration de WordPress, et dans le menu Apparence, ouvrez l'onglet Widgets. Prenez un ou plusieurs widgets dans la colonne de gauche et ajoutez-les dans celle de droite (voir Figure 6.31).



Figure 6.31
Ajout de widgets dans la colonne latérale.

Enregistrez les modifications et ouvrez une nouvelle fois la page d'accueil de votre blog : les éléments que vous avez insérés auparavant ont disparu pour laisser place aux widgets que vous venez d'ajouter (voir Figure 6.32).



Figure 6.32
Les widgets apparaissent dans la colonne latérale.

Deuxième méthode

Il existe pour créer vos zones widgetisables une autre manière qui est un peu plus compliquée mais qui vous permettra notamment de nommer ces différentes zones mais aussi de choisir le type de zone que vous voulez.

Prenons un exemple : imaginons que vous souhaitiez créer une zone widgetisable dans votre colonne latérale, y mettre une liste avec un titre H3 et que cette liste corresponde aux derniers commentaires.

Dans votre colonne latérale, vous allez devoir entrer le code suivant :

```
<?php if ( function_exists ( 'dynamic_sidebar' ) && dynamic_
sidebar("commentaires") ) : ?>
```

On a ajouté le nom de la zone widgetisable. Ensuite, vous allez entrer le code suivant dans le fichier functions.php :

```
<?php
if ( function_exists('register_sidebar') )
    register_sidebar(array(
  'name' => __( 'Commentaires' ),
        'before_widget' => '<li>',
        'after_widget' => '</li>',
        'before_title' => '<h3">',
        'after_title' => '</h3>',
    ));
?>
```

Ici, on a une fonction plus complète qui va vous permettre d'entrer plus d'informations sur la zone que vous souhaitez créer. Tout d'abord, vous allez y mettre le nom de la zone à widgetiser pour que WordPress fasse la relation avec le code que vous avez entré dans le fichier sidebar.php. Ensuite, vous décidez du type de contenu que vous souhaitez mettre dans cette zone. Ici, on a choisi une liste de commentaires dont on utilise les balises "li". Mais vous pouvez très bien choisir un autre type de contenu. En dessous vient le type de titre que l'on va utiliser pour le titre. Ici, on choisit les titres de type H3.

Une fois que vous avez saisi toutes ces informations dans les bons fichiers, vous pouvez aller sur l'administration de votre blog, au niveau des widgets et, en haut à droite, vous voyez apparaître votre zone widgetisable, tout juste créée. Et, au lieu d'avoir "colonne latérale 1", vous voyez s'afficher le nom que vous avez choisi (voir Figure 6.33). Cela permet de mieux gérer toutes ces zones dès lors qu'elles sont nombreuses.



Figure 6.33
Création d'une zone widgetisable "Commentaires".

Insertion du pied de page

Les informations contenues dans le pied de page peuvent être différentes d'un blog à un autre. Cependant, quelques-unes se révèlent importantes et utiles :

- Mettez un copyright sur le contenu et le design de votre blog.

- Montrez que vous utilisez WordPress (un peu de publicité ne fait pas de mal, et cela peut être très utile pour les personnes visitant votre blog).

Ce sont les informations que nous allons proposer ici. Ouvrez donc le fichier ; le code à insérer sera le suivant :

```
<div id="footer">
  <p>Copyright &#169; <?php print(date(Y)); ?> <?php bloginfo('name'); ?>
    <br />
    Blog propulsé par <a href="http://wordpress.org/">WordPress</a> et
con&ccedil;u par <a href="http://www.fran6art.com">Fran6art</a>
  </p>
</div>
```

Ici, vous avez créé tout d'abord le bloc qui va envelopper l'ensemble du pied de page grâce à la balise `footer`. C'est un `id`, il est donc unique. Ensuite, vous avez créé un paragraphe (balise `p`) dans lequel vous avez inséré les informations de copyright, que vous avez associées à l'année en cours *via* un simple appel à la fonction PHP `date()`, et au nom du blog par le biais du marqueur de modèle `bloginfo()` que vous avez vu plusieurs fois déjà auparavant.

Enfin, vous avez ajouté une ligne avec quelques informations montrant que vous utilisez WordPress.

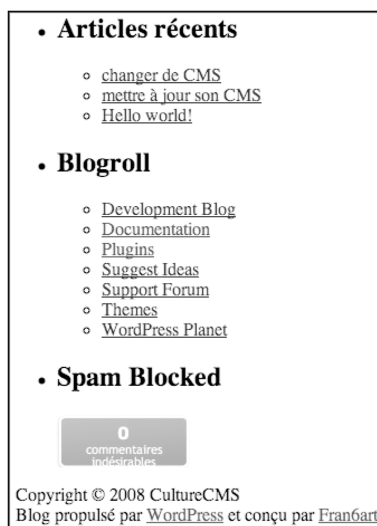
Enregistrez votre fichier en le nommant footer.php.

Maintenant, vous allez devoir insérer le marqueur de modèle pour le pied de page dans le fichier index.php. Directement sous le marqueur pour la colonne latérale, vous allez insérer le code suivant :

```
<?php get_footer(); ?>
```

Enregistrez votre fichier et rafraîchissez votre navigateur. Vous avez maintenant un pied de page qui affiche les informations que vous venez d'insérer (voir Figure 6.34).

Figure 6.34
Affichage du pied de page.



Pour terminer, vous allez placer le crochet (hook) pour les extensions directement sous le code inséré précédemment et avant la fermeture de la div `footer` :

```
    </p>
    <?php wp_footer(); ?>
  </div>
```

Création des fichiers pour les différents types de pages

Pour le moment, vous n'avez qu'un type de page, index.php. C'est donc elle qui est utilisée pour afficher toutes les pages du blog : accueil, article seul, page statique seule… Conformément à la hiérarchie des modèles WordPress, vous allez maintenant concevoir d'autres fichiers pour les différents types de pages. Vous ne créerez pas tous les fichiers proposés par la hiérarchie, mais uniquement les pages qui sont incontournables, à savoir :

- la page d'article (single.php) ;
- les pages statiques (page.php) ;
- la page d'archives, de catégories et de mots-clefs (archive.php) ;
- la page affichant les résultats de recherche (search.php).

Création de la page d'article

Si vous vous rendez sur la page d'accueil de votre blog et cliquez sur le titre d'un de vos articles, vous arrivez sur l'article seul, qui reprend le contenu du fichier index.php. Vous avez donc l'en-tête, le contenu de l'article, la colonne latérale et le pied de page. Il vous manque quelque chose de très important : les commentaires.

Vous allez créer un fichier que vous appellerez single.php, et vous y copierez le contenu du fichier index.php. Ici, il y a une seule différence entre l'affichage sur la page d'accueil et celle-ci : le nombre de commentaires n'apparaît plus dans les métadonnées de l'article. En effet, ce marqueur de modèle n'est pas fonctionnel sur la page d'article. Qui plus est, étant donné que vous êtes déjà sur la page d'article, cette notion n'a plus grand intérêt.

Vous allez donc retirer le code, ainsi que l'élément séparateur qui le précède, du fichier single.php, à savoir :

```
  | <?php comments_popup_link('Pas de commentaires', '1 Commentaire', '%
➥Commentaires'); ?>
```

Insertion des commentaires dans la page d'article

Nous avons vu dans les paramétrages des commentaires qu'il existe deux manières de les afficher :

- sous forme de liste linéaire, sans hiérarchie et uniquement dans le sens antéchronologique ;
- sous forme de conversation hiérarchique où il est possible de répondre à un commentaire en particulier, selon le nombre de niveaux définis.

Cette notion de conversation hiérarchisée est apparue avec la version 2.7 de WordPress. Avant, seule la première fonction était disponible.

Mais attention, la conversation hiérarchisée est proposée à partir de la version 2.7 de WordPress mais elle n'est pas activée par défaut. Nous en avons déjà parlé précédemment mais cela reste une option. Vous pouvez très bien afficher vos commentaires sous forme de liste ordonnée, sans niveau, comme c'était le cas auparavant.

Avec l'arrivée de cette fonctionnalité, le code des commentaires de WordPress a été modifié. Ainsi, de nombreux thèmes ont été créés avec l'ancien code, et d'autres plus récents ont été créés avec le nouveau code.

Ici, nous allons construire notre thème avec le nouveau code des commentaires, qui permet une gestion hiérarchique de la conversation, mais nous verrons également comment créer la partie "commentaires" avec l'ancien code.

En effet, si vous utilisez une version de WordPress antérieure à la 2.7, le nouveau code ne fonctionnera pas. Vous aurez donc besoin de l'ancien. Qui plus est, de nombreux thèmes ont été créés avant la sortie de la version 2.7 de WordPress et utilisent donc l'ancien code.

Par conséquent, que vous souhaitiez créer un thème pour une version de WordPress antérieure à la 2.7 ou que vous souhaitiez simplement comprendre comment fonctionnent les commentaires sur des thèmes conçus pour les versions précédentes de WordPress, il est important de détailler le code et de vous permettre de l'utiliser.

Création du fichier comments.php permettant les commentaires hiérarchisés (WordPress 2.7 et supérieures)

Les commentaires vont apparaître sur la page d'article de notre thème. Le fichier concerné est single.php. Vous allez donc ouvrir ce fichier et y insérer un marqueur de modèle pour appeler le fichier des commentaires, comments.php. En effet, comme nous l'avons vu au chapitre précédent, les commentaires font partie d'un fichier séparé de celui de l'article.

Le modèle utilisé sera `comments_template()`, et le code à insérer dans le fichier single.php est le suivant :

```
<div class="comments-template">
  <?php comments_template(); ?>
</div>
<?php endwhile; ?>
```

Vous l'ajoutez directement au-dessus de la commande `endwhile`. Nous utilisons une balise avec une classe que nous appelons `comments-template` et le modèle `comments_template()` pour afficher les commentaires sur la page d'article.

Créez maintenant un nouveau fichier, que vous appellerez comments.php. Dans ce fichier, vous ajouterez toutes les informations qui permettront d'afficher les commentaires mais aussi le formulaire de réponse.

Commencez par insérer le code suivant :

```
<?php // Do not delete these lines
if (!empty($_SERVER['SCRIPT_FILENAME']) && 'comments.php' == basename($_
➥SERVER['SCRIPT_FILENAME']))
```

```
    die ('Merci de ne pas lancer cette page directement.');

  if ( post_password_required() ) { ?>
      <p class="nocomments">Cet article est protégé par un mot de passe. Entrez
  ce mot de passe pour lire les commentaires.</p>
      <?php
      return;
    }
  ?>
```

Ces premières lignes sont utilisées pour la gestion des articles par mot de passe. Si un article est protégé par un mot de passe, les commentaires ne s'afficheront pas et le message suivant apparaîtra à la place : "Cet article est protégé par un mot de passe. Entrez ce mot de passe pour lire les commentaires."

Ensuite, ajoutez la boucle des commentaires sous WordPress :

```
<!-- You can start editing here. -->

<?php if ($comments) : ?>

  <h3 id="comments"><?php comments_number('Aucun commentaire',
 'Un commentaire', '% commentaires' );?> pour &#8220;<?php the_title();?>
 &#8221;</h3>

  <ol class="commentlist">
  <?php wp_list_comments(); ?>
  </ol>

  <div class="navigation">
    <div class="alignleft"><?php previous_comments_link() ?></div>
    <div class="alignright"><?php next_comments_link() ?></div>
  </div>
 <?php else : // this is displayed if there are no comments so far ?>

  <?php if ('open' == $post->comment_status) : ?>
    <!-- If comments are open, but there are no comments. -->

  <?php else : // comments are closed ?>
    <!-- If comments are closed. -->
    <p class="nocomments">Les commentaires sont ferm&eacute;s.</p>

  <?php endif; ?>
<?php endif; ?>
```

Voyons dans le détail le contenu de ces quelques lignes de code. La structure de la boucle des commentaires a la forme suivante :

```
<?php if ($comments) : ?>
```

S'il y a des commentaires, nous allons les afficher. Mais tout d'abord nous afficherons le nombre de commentaires pour l'article en question :

```
<h3 id="comments"><?php comments_number('Aucun commentaire', 'Un commentaire',
'% commentaires' );?> pour &#8220;<?php the_title(); ?>&#8221;</h3>
```

Ensuite, nous afficherons les commentaires par le biais du marqueur de modèle `wp_list_comments()` apparu avec la version 2.7 de WordPress :

```
<ol class="commentlist">
  <?php wp_list_comments(); ?>
  </ol>
```

Ce marqueur englobe toute une série d'informations comme :

- l'avatar du commentateur ;
- le nom du commentateur ;
- la date de commentaire ;
- le contenu du commentaire ;
- un lien pour répondre au commentaire si l'option a été activée (voir Chapitre 3).

WordPress, depuis sa version 2.7, permet également de choisir le nombre de commentaires à afficher par page. Ici, nous allons insérer des liens permettant de naviguer sur les différentes pages de commentaires :

```
<div class="navigation">
    <div class="alignleft"><?php previous_comments_link() ?></div>
    <div class="alignright"><?php next_comments_link() ?></div>
  </div>
```

Ensuite, nous ajouterons le contenu dans le cas où il n'y aurait pas de commentaires ou que ceux-ci seraient fermés :

```
<?php else : // this is displayed if there are no comments so far ?>

  <?php if ('open' == $post->comment_status) : ?>
    <!-- If comments are open, but there are no comments. -->

  <?php else : // comments are closed ?>
    <!-- If comments are closed. -->
    <p class="nocomments">Les commentaires sont ferm&eacute;s.</p>

  <?php endif; ?>
<?php endif; ?>
```

S'il n'y a pas de commentaires, rien ne s'affiche, et si les commentaires sont fermés, alors la phrase suivante apparaît : "Les commentaires sont fermés."

Maintenant que la boucle des commentaires est insérée, nous allons nous intéresser au formulaire qui permettra au visiteur de laisser son propre commentaire. Tout d'abord, vous allez ajouter les lignes suivantes, directement sous la boucle :

```
<?php if ('open' == $post->comment_status) : ?>
<div id="respond">

<h3><?php comment_form_title( 'Laisser un commentaire', 'Laisser un commentaire
&agrave; %s' ); ?></h3>
```

```
<div class="cancel-comment-reply">
  <small><?php cancel_comment_reply_link(); ?></small>
</div>
```

La première ligne indique que, si les commentaires sont ouverts, nous allons afficher le formulaire et donc insérer ce qui suit.

Tout d'abord, vous ouvrez la balise `respond` qui va englober toute la partie "réponse" des commentaires, c'est-à-dire principalement le formulaire.

Comme ici, vous pourrez laisser une réponse à un commentaire précis. Vous avez deux possibilités pour présenter le formulaire. "Laisser un commentaire" s'affiche si vous répondez à la suite d'un commentaire, mais sans le faire directement. Et "Laisser un commentaire à" apparaît si vous répondez directement à un commentaire en particulier.

Ensuite, vous affichez le marqueur de modèle `cancel_comment_reply_link()` qui permettra de refermer le formulaire de contact prévu pour répondre à un commentaire précis.

Visuellement, lorsque vous répondez à un commentaire précis, un formulaire de contact s'ouvre directement sous ce commentaire (voir Figure 6.35). Cette possibilité est offerte par JavaScript.



Figure 6.35
Affichage du formulaire de réponse directement sous le commentaire.

Et, à ce niveau-là, un lien vous permet de refermer le formulaire de réponse au commentaire.

Ensuite, vous allez insérer quelques lignes de code qui seront utiles pour les blogs qui doivent obligatoirement être connectés pour laisser un commentaire :

```
<?php if ( get_option('comment_registration') && !$user_ID ) : ?>
<p>Vous devez &ecirc;tre <a href="<?php echo get_option('siteurl'); ?>/wp-
login.php?redirect_to=<?php echo urlencode(get_permalink());
?>">connect&eacute;</a> pour publier un commentaire.</p>
<?php else : ?>
```

```
<form action="<?php echo get_option('siteurl'); ?>/wp-comments-post.php"
method="post" id="commentform">

<?php if ( $user_ID ) : ?>

<p>Connect&eacute; en tant que <a href="<?php echo get_option('siteurl'); ?>/
wp-admin/profile.php"><?php echo $user_identity; ?></a>. <a href="<?php echo
wp_logout_url(get_permalink()); ?>" title="Log out of this account">Se
d&eacute;connecter  &raquo;</a></p>
```

En fait, ici, il y a une condition qui dit : "Si l’enregistrement est obligatoire pour laisser un commentaire, alors le visiteur doit se connecter, sinon on affiche directement le formulaire de commentaire."

Si l’utilisateur est connecté, différentes informations vont apparaître. Il aura notamment la possibilité de se déconnecter. S’il n’est pas connecté, nous allons afficher le formulaire complet de réponse. Pour cela, vous allez ajouter le code suivant :

```
<?php else : ?>

<p><input type="text" name="author" id="author" value="<?php echo $comment_
author; ?>" size="22" tabindex="1" <?php if ($req) echo "aria-required='true'";
?> />
<label for="author"><small>Nom <?php if ($req) echo "(required)"; ?></small>
</label></p>

<p><input type="text" name="email" id="email" value="<?php echo $comment_
author_email; ?>" size="22" tabindex="2" <?php if ($req) echo "aria-
required='true'"; ?> />
<label for="email"><small>Adresse e-mail (ne sera pas publi&eacute;e)
<?php if ($req) echo "(required)"; ?></small></label></p>

<p><input type="text" name="url" id="url" value="<?php echo $comment_author_
url; ?>" size="22" tabindex="3" />
<label for="url"><small>Site Web</small></label></p>

<?php endif; ?>

<!--<p><small><strong>XHTML:</strong> You can use these tags: <code><?php echo
allowed_tags(); ?></code></small></p>-->

<p><textarea name="comment" id="comment" cols="100%" rows="10" tabindex="4">
</textarea></p>

<p><input name="submit" type="submit" id="submit" tabindex="5" value="Dites-le
!" />
<?php comment_id_fields(); ?>
</p>
<?php do_action('comment_form', $post->ID); ?>

</form>
</div>
```

```
<?php endif; // If registration required and not logged in ?>

<?php endif; // if you delete this the sky will fall on your head ?>
```

Comme pour la boucle des commentaires, nous allons prendre ici le temps de détailler ces quelques lignes.

Nous allons tout d'abord afficher les différentes lignes du formulaire reprenant le nom, l'adresse e-mail et le site web, tout en leur donnant une taille :

```
<?php else : ?>

<p><input type="text" name="author" id="author" value="<?php echo $comment_
author; ?>" size="22" tabindex="1" <?php if ($req) echo "aria-required='true'";
?> />
<label for="author"><small>Nom <?php if ($req) echo "(required)"; ?></small>
</label></p>

<p><input type="text" name="email" id="email" value="<?php echo $comment_
author_email; ?>" size="22" tabindex="2" <?php if ($req) echo "aria-
required='true'"; ?> />
<label for="email"><small>Adresse e-mail (ne sera pas publi&eacute;e) <?php if
($req) echo "(required)"; ?></small></label></p>

<p><input type="text" name="url" id="url" value="<?php echo $comment_author_
url; ?>" size="22" tabindex="3" />
<label for="url"><small>Site Web</small></label></p>

<?php endif; ?>
```

Ensuite, nous avons la possibilité d'insérer une ligne qui permettra au commentateur de savoir quels marqueurs il peut utiliser dans le formulaire (voir Figure 6.36).

```
<!--<p><small><strong>XHTML:</strong> You can use these tags: <code><?php echo
allowed_tags(); ?></code></small></p>-->
```

XHTML: You can use these tags: `<a href="" title=""> <abbr title=""> <acronym title=""> <b> <blockquote cite=""> <cite> <code> <del datetime=""> <em> <i> <q cite=""> <strike> <strong>`

Figure 6.36
Marqueurs utilisables dans le formulaire de commentaire.

Cependant, cette fonctionnalité n'est pas activée puisqu'elle est présentée comme commentaire. Pour l'activer, il suffit de retirer les `<!--` et `-->` situés avant et après le code.

Enfin, nous affichons la zone de saisie du texte, ainsi que le bouton pour valider son commentaire, et nous refermons l'ensemble du formulaire, ainsi que les différentes conditions que nous avons ouvertes précédemment.

```
<p><textarea name="comment" id="comment" cols="100%" rows="10" tabindex="4">
</textarea></p>

<p><input name="submit" type="submit" id="submit" tabindex="5" value="Dites-
le !" />
<?php comment_id_fields(); ?>
```

```
</p>
<?php do_action('comment_form', $post->ID); ?>

</form>
</div>

<?php endif; // If registration required and not logged in ?>

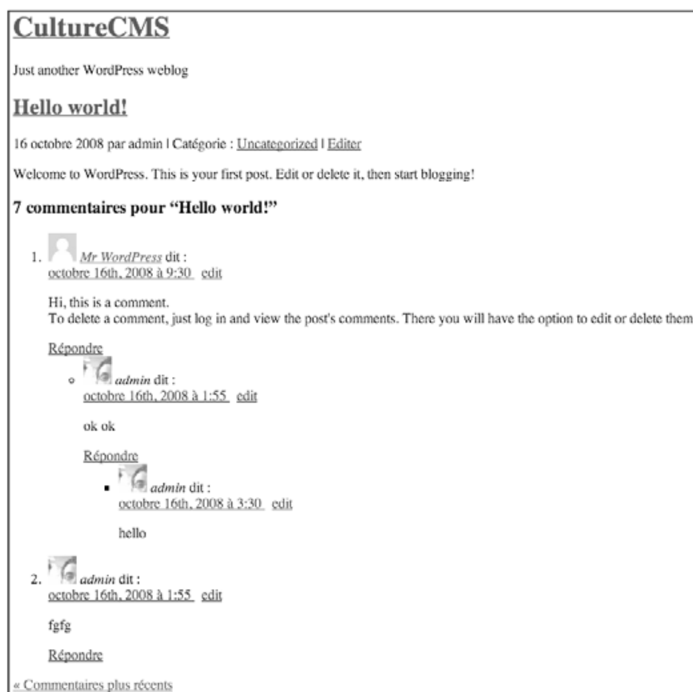
<?php endif; // if you delete this the sky will fall on your head ?>
```

Nous avons vu un peu plus haut que le formulaire de réponse directe allait se positionner directement sous le commentaire en utilisant du JavaScript. Cette option ne sera effective que si vous ajoutez un morceau de code dans votre fichier header.php, juste au-dessus du hook `wp_head()` :

```
if ( is_singular() ) wp_enqueue_script( 'comment-reply' );
```

Une fois que vous avez inséré ce code, enregistrez et fermez votre fichier header.php. Faites de même pour votre fichier comments.php. Ouvrez alors votre navigateur et allez sur la page de votre blog. La zone réservée aux commentaires apparaît désormais sous le contenu de vos articles (voir Figure 6.37).

Figure 6.37
Affichage des commentaires, liste hiérarchisée.



Création du fichier comments.php pour la version 2.6 de WordPress et inférieures

Ici, pas de commentaires hiérarchisés, nous allons simplement créer une liste ordonnée de commentaires. Le principe est quasi le même que pour la version 2.7 du fichier comments.php, mais à quelques différences près qui ont leur importance. Voici donc en détail le contenu de ce fichier pour la version 2.6 de WordPress ainsi que les versions précédentes.

Tout d'abord, insérez les lignes de code suivantes :

```
<?php // Do not delete these lines
if (!empty($_SERVER['SCRIPT_FILENAME']) && 'comments.php' == basename
($_SERVER['SCRIPT_FILENAME']))
  die ('Merci de ne pas lancer cette page directement.');

if (!empty($post->post_password)) { // if there's a password
  if ($_COOKIE['wp-postpass_' . COOKIEHASH] != $post->post_password)
{  // and it doesn't match the cookie
?>
<p class="nocomments">Cet article est protégé par un mot de passe. Entrez ce
mot de passe pour lire les commentaires.</p>
?php
  return;
  }
}

/* This variable is for alternating comment background */
$oddcomment = 'class="alt" ';
?>
```

Ces quelques lignes vont nous renseigner sur :

- le fait que ce fichier n'est pas lancé directement, mais bien aussi d'une autre page ;
- la vérification du mot de passe pour les articles protégés ;
- la mise en place d'une variable contenant la classe CSS qui nous permettra au besoin de styler différemment les commentaires pairs et impairs.

Ensuite, insérez le code suivant :

```
<?php if ($comments) : ?>
```

Cette ligne stipule que s'il y a des commentaires, il faut déclencher le code contenu dans le bloc de cette condition. Voici le contenu de ce bloc :

```
<h3 id="comments"><?php comments_number('Aucun commentaire', 'Un commentaire',
'% commentaires' );?> pour &#8220;<?php the_title(); ?>&#8221;</h3>
```

Vous affichez un titre en `h3`, qui va précéder l'ensemble des commentaires et qui va dire combien il y a de commentaires – marqueur de modèle `comments_number()` – pour l'article en question – marqueur de modèle `the_title()`. Maintenant nous allons afficher la liste complète de ces commentaires :

```
<ol class="commentlist">
  <?php foreach ($comments as $comment) : ?>
    <?php $comment_type = get_comment_type(); ?>
```

```
        <?php if($comment_type == 'comment') { ?>
          <li <?php echo $oddcomment; ?>id="comment-<?php comment_ID() ?>">
            <div class="comment_author"><?php echo get_avatar( $comment, 32 ); ?>
            <div class="comment_info"><cite><?php comment_author_link() ?></cite>
            <?php if ($comment->comment_approved == '0') : ?>
              <em>Votre commentaire est en attente de mod&eacute;ration.</em>
            <?php endif; ?>
            <br />
            <small class="commentmetadata"><a href="#comment-<?php comment_ID() ?>
            " title=""><?php comment_date('j F Y') ?> à <?php comment_time() ?></a>
            <?php edit_comment_link('Editer','-- ',''); ?></small>
          <?php comment_text() ?>
        </li>
        <?php /* Changes every other comment to a different class */ $oddcomment
        = ( empty( $oddcomment ) ) ? 'class="alt" ' : ''; ?>
      <?php } /* End of is_comment statement */ ?>
    <?php endforeach; /* end for each comment */ ?> </ol>
```

Ces commentaires apparaissent sous forme de liste ordonnée (balise `ol`), et pour chacun d'eux, WordPress affichera :

- L'identifiant du commentaire, `comment_ID()`.
- L'avatar du commentateur, c'est-à-dire une image liée à son adresse e-mail, `get_avatar()`.
- Le nom de l'auteur du commentaire, avec un lien vers son blog si celui-ci a été précisé, `comment_author_link()`.
- Un test pour voir si le commentaire n'a pas encore été approuvé par le propriétaire du blog (`$comment->comment_approved == '0'`). Si c'est effectivement le cas, alors le texte suivant s'affichera : "Votre commentaire est en attente de modération."
- Les données suivantes :
  - sa date de publication, `comment_date()` ;
  - un lien direct pour le modifier, `edit_comment_link()` ;
  - enfin, le commentaire lui-même, `comment_text()`.

Chacune de ces informations est proprement balisée avec des classes explicites, afin de pouvoir facilement toutes les styler *via* CSS.

Ensuite, nous insérons un code qui va nous permettre de séparer les commentaires des rétroliens :

```
<ol class="trackbacks-layout">
  <?php foreach ($comments as $comment) : ?>
    <?php $comment_type = get_comment_type(); ?>
    <?php if($comment_type != 'comment') { ?>
      <li><?php comment_author_link() ?></li>
    <?php } ?>
  <?php endforeach; ?>
</ol>
```

Il s'agit là d'une seconde boucle qui parcourt cette fois uniquement les commentaires de types autres (donc rétroliens et pings) pour leur donner un traitement particulier. Ceux-ci ne verront affiché que le nom du blog d'où ils proviennent (le nom de l'auteur dans ce contexte).

S'il n'y a pas de commentaires, alors WordPress n'affiche rien ; dans le cas où les commentaires sont fermés, il affiche le texte "Les commentaires sont fermés".

```
<?php else : // this is displayed if there are no comments so far ?>
  <?php if ('open' == $post->comment_status) : ?>
    <!-- If comments are open, but there are no comments. -->
  <?php else : // comments are closed ?>
    <!-- If comments are closed. -->
    <p class="nocomments">Les commentaires sont ferm&eacute;s. </p>
  <?php endif; ?>
<?php endif; ?>
```

Il faut ensuite afficher le formulaire pour laisser un commentaire. Ici, le fichier prend en compte les blogs sur lesquels un visiteur ne peut laisser un commentaire que s'il est connecté et inscrit comme utilisateur du blog, ou alors tout simplement s'il est le rédacteur du blog. Ainsi, si l'auteur du blog a décidé de laisser ses commentaires ouverts à tous ou seulement aux membres inscrits, votre thème pourra prendre son choix en compte.

```
<?php if ('open' == $post->comment_status) : ?>
  <h3 id="respond">Laisser un commentaire</h3>
  <?php if ( get_option('comment_registration') && !$user_ID ) : ?>
    <p>Vous devez être <a href="<?php echo get_option('siteurl'); ?>/wp-login.
php?redirect_to=<?php echo urlencode(get_permalink()); ?>">connect&eacute;</a>
pour publier un commentaire.</p>
  <?php else : ?>
    <form action="<?php echo get_option('siteurl'); ?>/wp-comments-post.php"
method="post" id="commentform">
```

Si l'utilisateur est connecté, il verra ses informations de connexion affichées : son identifiant public, avec lequel il va signer son commentaire, et l'adresse de son site si elle existe. Il pourra aussi choisir de se déconnecter.

```
<?php if ( $user_ID ) : ?>
  <p>Connect&eacute; en tant que <a href="<?php echo get_option('siteurl'); ?>/
wp-admin/profile.php"><?php echo $user_identity; ?></a>. <a href="<?php echo
get_option('siteurl'); ?>/wp-login.php?action=logout" title="Se déconnecter du
site.">Se d&eacute;connecter  &raquo;</a></p>
```

S'il n'est pas connecté, il pourra saisir son nom, son adresse e-mail et l'adresse de son site web :

```
<?php else : ?>
    <p><input type="text" name="author" id="author" value="<?php echo $comment_
author; ?>" size="22" tabindex="1" />
<label for="author"><small>Nom <?php if ($req) echo "(obligatoire)"; ?></
small></label></p>
    <p><input type="text" name="email" id="email" value="<?php echo $comment_
author_email; ?>" size="22" tabindex="2" />
    <label for="email"><small>Adresse e-mail (ne sera pas publié) <?php if
($req) echo "(obligatoire)"; ?></small></label></p>
```

```
    <p><input type="text" name="url" id="url" value="<?php echo $comment_
author_url; ?>" size="22" tabindex="3" />
    <label for="url"><small>Site Web</small></label></p>
<?php endif; ?>
```

Vous avez la possibilité d'afficher une liste de balises XHTML qui seront autorisées dans la rédaction du commentaire. Vous retrouvez ensuite la zone de saisie du commentaire et, enfin, le bouton de validation du commentaire :

```
<!--<p><small><strong>XHTML:</strong> Vous pouvez utiliser ces tags:
<code><?php echo allowed_tags(); ?></code></small></p>-->
<p><textarea name="comment" id="comment" cols="100%" rows="10" tabindex="4">
</textarea></p>
<p><input name="submit" type="submit" id="submit" tabindex="5" value="Dites-le
!" />
<input type="hidden" name="comment_post_ID" value="<?php echo $id; ?>" /></p>
<?php do_action('comment_form', $post->ID); ?>
</form>
<?php endif; // If registration required and not logged in ?>
<?php endif; // if you delete this the sky will fall on your head ?>
```

Une fois le fichier comments.php du thème default copié dans le dossier de votre propre thème, rafraîchissez la fenêtre de votre navigateur. La partie consacrée aux commentaires apparaît effectivement sous votre article et avant le contenu de la colonne latérale. Si vous avez déjà quelques commentaires, vous pouvez les voir apparaître (voir Figure 6.38).



Figure 6.38

Affichage des commentaires, liste ordonnée.

Création de la page d'archives, de catégories et de mots-clefs

Le fichier archive.php va non seulement permettre d'afficher une page d'archives, mais il sera également utilisé pour les pages des catégories et des mots-clefs.

Pour cela, nous allons prendre la hiérarchie WordPress à rebours et utiliser une version adaptée du fichier index.php. Vous allez reprendre le contenu du fichier index.php, mais vous le modifierez légèrement pour ne faire apparaître que le début de l'article.

Vous allez donc créer un nouveau fichier, archive.php, et y coller tout le contenu du fichier index.php. Ensuite, vous chercherez la boucle WordPress et dans celle-ci vous remplacerez :

```
<?php the_content(); ?>
```

par

```
<?php the_excerpt(); ?>
```

Ici, tout le contenu de l'article, `the_content()`, sera remplacé par ses 55 premiers mots, `the_excerpt()`.

Enregistrez le fichier et allez sur la page d'accueil de votre blog. Cliquez sur une des catégories présentes dans la colonne latérale : vous voyez désormais une page avec les premières lignes des articles et non plus leur contenu en entier (voir Figure 6.39).

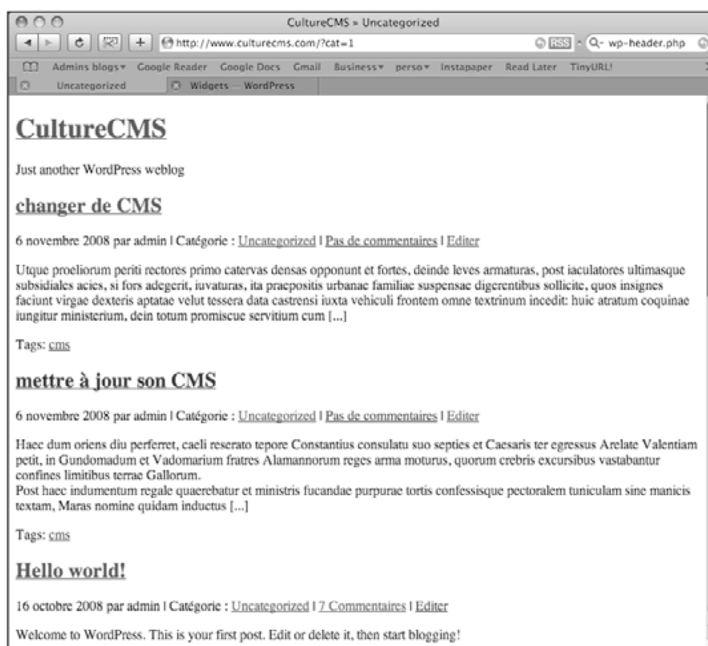


Figure 6.39

Affichage de la page d'archives.

Création de la page d'affichage des résultats de recherche

Vous allez créer un nouveau fichier, search.php, et copier son contenu au-dessus, à savoir dans archive.php – lui-même déjà adapté de index.php. Ceux-ci ne sont pas liés dans la

hiérarchie des fichiers WordPress. De ce fait, si vous ne créez pas de fichier search.php, c'est index.php qui sera utilisé.

Création du fichier pour afficher les pages statiques du blog

Ici, une fois de plus, vous allez utiliser le contenu du fichier index.php, tout en retirant quelques informations qui ne sont pas très utiles pour ce type de page, à savoir les métadonnées pour la page.

Créez donc le fichier page.php et copiez-y le contenu de l'index. Ensuite, enlevez les métadonnées :

```
<p class="postmetadata">
    <?php the_time('j F Y'); ?> par <?php the_author(); ?> |
    Cat&eacute;gorie: <?php the_category(', '); ?> |
    <?php comments_popup_link('Pas de commentaires', '1 Commentaire', '%
Commentaires'); ?> <?php edit_post_link('Editer', ' &#124; ', ''); ?>
  </p>
```

Enfin, sous le `php endwhile`, et avant le `php endif`, insérez le code suivant :

```
<?php edit_post_link('Modifier cette page', '<p>', '</p>'); ?>
```

Ce code va ajouter une ligne, sous l'article, qui vous permettra de modifier son contenu directement, comme la fonction "edit", au niveau de chaque article.

Enregistrez le fichier, et rafraîchissez le navigateur. Maintenant, les autres pages de votre blog apparaissent sans les métadonnées (voir Figure 6.40).

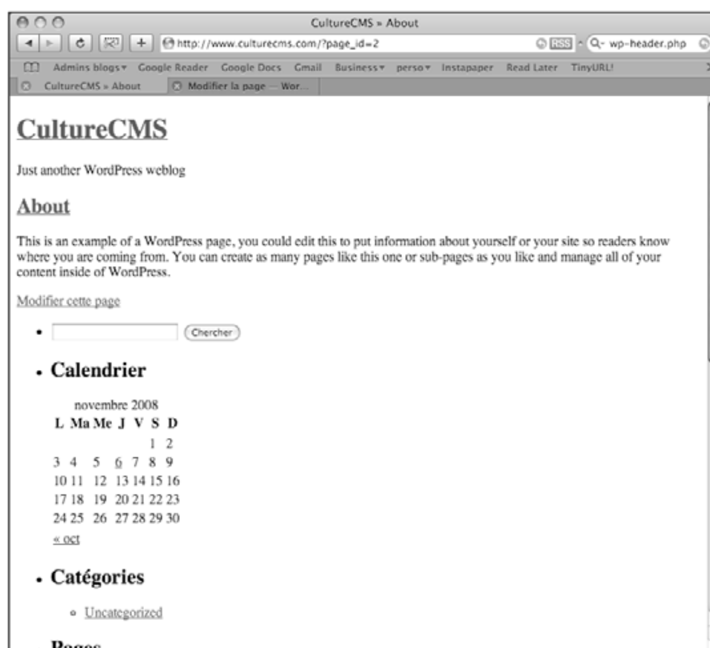


Figure 6.40

Affichage d'une page statique.

Création de la page 404

La page 404 s'affiche lorsque la requête du visiteur ne mène à rien – soit que l'article demandé a été effacé/fermé, soit que l'adresse saisie n'existe tout simplement pas. Le visiteur reçoit alors un message l'informant que ce qu'il cherche n'existe pas.

Dans cette page, vous allez insérer la structure du blog, à savoir l'en-tête, la colonne latérale et le pied de page, et vous remplacerez le contenu par une petite phrase simple du style "Désolé, mais vous cherchez quelque chose qui ne se trouve pas ici", accompagnée d'un formulaire de recherche pour aider le visiteur à trouver ce qu'il cherche sur votre blog.

Commencez par créer un fichier 404.php. Dans ce fichier, copiez le fichier index.php, puis retirez le contenu de la balise `content`, c'est-à-dire la boucle WordPress :

```
<?php if(have_posts()) : ?><?php while(have_posts()) : the_post(); ?>
  <div class="post" id="post-<?php the_ID(); ?>">
    <h2><a href="<?php the_permalink(); ?>" title="<?php the_title(); ?>">
<?php the_title(); ?></a></h2>
    <p class="postmetadata">
      <?php the_time('j F Y'); ?> par <?php the_author(); ?> | Cat&eacute;gorie:
<?php the_category(','); ?> | <?php comments_popup_link('Pas de commentaires',
'1 Commentaire', '% Commentaires'); ?><?php edit_post_link('Editer', ' &#124;
', ''); ?>
    </p>
    <div class="post_content">
      <?php the_content(); ?>
    </div>
    <p class="tags"><?php the_tags(); ?></p>
  </div>
<?php endwhile; ?>
<?php else : ?>
  <h2 class="center">Introuvable</h2>
  <p class="center">D&eacute;sol&eacute;, mais vous cherchez quelque chose qui
ne se trouve pas ici .</p>
  <div id="searchno"><?php include (TEMPLATEPATH . "/searchform.php"); ?></div>
<?php endif; ?>
```

Et remplacez toute cette section par le code suivant, qui comprend une phrase courte expliquant que WordPress n'a rien trouvé en rapport avec la demande du visiteur, ainsi que notre formulaire de recherche :

```
<p>Désolé, mais vous cherchez quelque chose qui ne se trouve pas ici .</p>

<?php include (TEMPLATEPATH . "/searchform.php"); ?>
```

Ouvrez maintenant votre navigateur et tapez l'URL de votre blog avec quelques lettres aléatoires à la fin, par exemple *www.monblog.com/fgrggreg*. Normalement, WordPress ne va rien trouver et affichera le contenu de la page 404 (voir Figure 6.41).



Figure 6.41
Affichage de la page 404.

Navigation entre les pages de résultats

Il est toujours important de proposer à un visiteur qui arrive sur la page d'accueil, ou sur une page de catégories, par exemple, un lien en bas de chaque page lui permettant d'aller voir le reste des articles ou des résultats.

Vous allez donc insérer un morceau de code qui permettra de naviguer de page en page. Ce code devra être placé sur les fichiers suivants :

- index.php ;
- archive.php ;
- search.php.

Ce code est à insérer après la fin de l'affichage des articles dans la boucle, à savoir `<?php endwhile; ?>` :

```
<div class="navigation">
  <div class="alignleft"><?php next_posts_link('&laquo; Articles plus anciens')
?></div>
  <div class="alignright"><?php previous_posts_link('Articles plus récents
&raquo;') ?></div>
</div>
```

Validation XHTML du thème

Votre thème est désormais terminé pour ce qui concerne le code XHTML et PHP. Avant de passer à la personnalisation du thème et à sa mise en forme, vous allez devoir valider votre thème auprès du W3C (www.w3.org), qui est en quelque sorte l'organisme chargé de promouvoir les standards du Web.

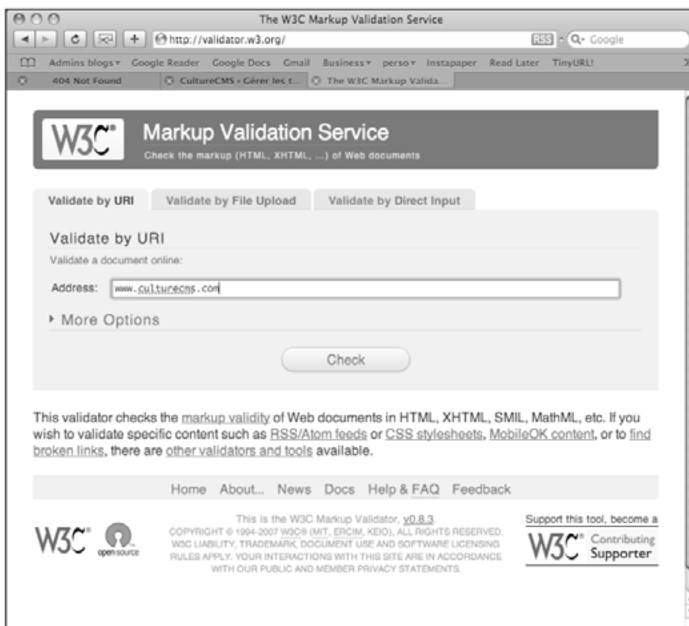
Si vous travaillez en local, vous devez récupérer le code source pour effectuer la validation. Chaque navigateur est censé proposer le code source dans un de ses onglets. Par exemple, sous Firefox, cela se passe sous l'onglet Affichage, puis Code source de la page.

Rendez-vous donc sur la page web de votre blog et récupérez le code source. Vous le copierez puis irez sur le site de validation du W3C, à l'adresse http://validator.w3.org.

Si votre blog est déjà en ligne, vous n'aurez qu'à soumettre son URL à la même adresse (voir Figure 6.42).

Figure 6.42

Soumission du blog à la validation du W3C.

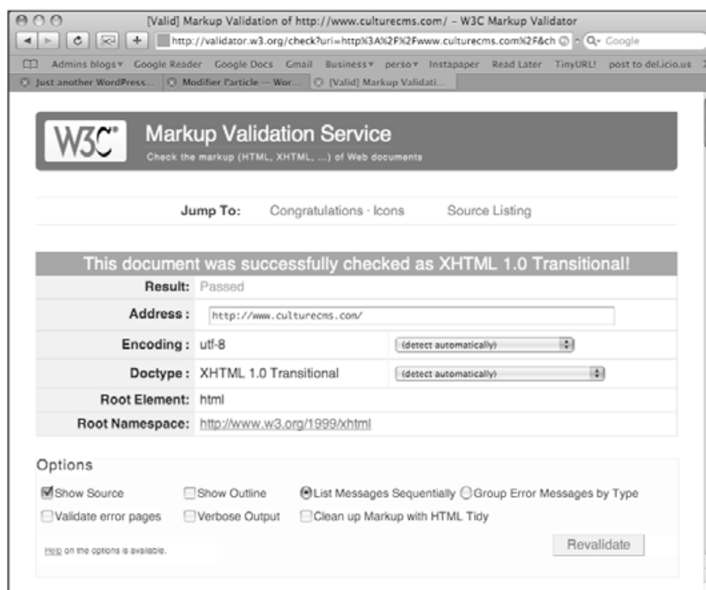


Sur le site, vous avez trois manières différentes de soumettre votre blog : soit par l'URL si votre thème est en ligne, soit en joignant un fichier, soit en soumettant le code source directement. Choisissez la méthode la plus appropriée à votre situation et soumettez votre code à validation en appuyant sur le bouton Check.

Si vous avez bien suivi l'ensemble de ce chapitre, la fenêtre suivante apparaît (voir Figure 6.43).

Figure 6.43

Validation XHTML de votre blog.



Le code est donc valide et vous êtes maintenant prêt à vous attaquer à la mise en forme de votre blog. Dans le cas contraire, récupérez directement sur **http://apprendre-wordpress.fr/livre** les différents fichiers du thème creation_theme.

Fonctions et options avancées de WordPress

WordPress regorge de fonctionnalités et de possibilités. C'est d'ailleurs pour cela que beaucoup de webdesigners adorent la plate-forme. Dans cette section, vous allez découvrir quelques fonctions dites "avancées" de WordPress, dont certaines ont déjà été évoquées plus tôt. Elles vous permettront de pousser encore un peu plus loin la personnalisation de votre blog et de votre thème. Celles-ci ne sont qu'un échantillon des possibilités de WordPress et la plupart restent en relation directe avec la création d'un article ou d'une page.

Fonction "more"

Le but de la fonction "more" est de permettre un affichage partiel de l'article sur la page d'accueil de votre blog (voir Figure 6.44). Il vous suffit d'utiliser le bouton prévu à cet effet dans les éditeurs visuels et HTML, de l'appliquer à l'endroit où vous souhaitez "couper" l'article sur la page et de n'insérer qu'une partie du texte sur la page d'accueil.

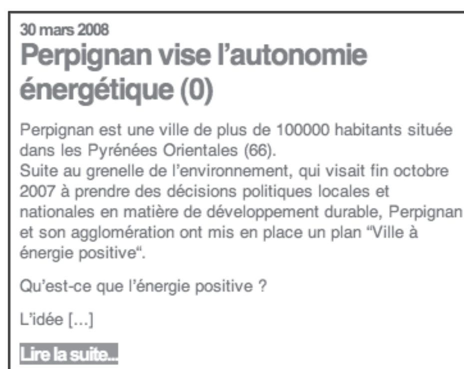


Figure 6.44
Exemple de fonction "more".

Paramétrage du thème pour utiliser la fonction "more"

Si la fonction est utilisable sans paramétrages avec le thème par défaut, il risque d'en être autrement avec un certain nombre d'autres thèmes. Il est donc important de savoir comment fonctionne cette option.

Nous partons du principe que nous utilisons le thème default de WordPress. Nous allons donc chercher le fichier qui va être employé ici pour faire apparaître les articles "tronqués" sur la page d'accueil du blog. Il s'agit bien du fichier index.php.

Rendez-vous sur votre hébergement et sous le dossier themes, ouvrez celui qui s'appelle default et trouvez-y le fichier index.php que vous allez afficher dans votre éditeur de texte.

Vous allez y chercher le code suivant :

```
<div class="entry">
  <?php the_content('Lire le reste de cet article &raquo;'); ?>
</div>
```

Dans ce code, nous retrouvons le marqueur `php the_content`, qui a pour objectif, comme son nom l'indique, d'afficher le contenu. Derrière, entre parenthèses, nous avons le texte qui va s'afficher dès que nous utiliserons la fonction "more".

Vous savez dorénavant où modifier le texte qui s'affichera en bas de vos articles "tronqués" sur la page d'accueil de votre blog.

Ensuite, la fonction "more" ne sera effective que si votre thème emploie le marqueur `php the_content`. Certains thèmes utilisent le marqueur `php the_excerpt`. Celui-ci ressemble beaucoup à la fonction "more" puisqu'il va, lui aussi, proposer un article tronqué en page d'accueil. Cependant, dans ce cas précis, vous ne pourrez pas choisir où vous souhaitez couper le texte. WordPress le fera automatiquement au 55e mot.

Par conséquent, si vous souhaitez utiliser la fonction "more", vérifiez bien au préalable que votre thème est correctement paramétré.

Notez également que la fonction "more" peut être utilisée sur tous les fichiers sauf sur les permaliens du blog, c'est-à-dire les pages d'articles.

Afficher un extrait d'article

Le concept ici est quelque peu semblable à l'utilisation de la fonction "more". Sur la page de rédaction d'un article, vous avez, dans les options avancées, un champ qui vous permet d'ajouter un "extrait" (voir Figure 6.45).

Extrait

Les extraits sont des résumés facultatifs de vos articles, écrits à la main. Vous pouvez les utiliser dans votre thème

Figure 6.45
Champ pour insérer un extrait d'article.

Dans ce champ, vous pouvez rédiger un petit résumé de votre article, quelques phrases qui vont synthétiser le contenu ou alors le présenter. Et c'est ce texte qui s'affichera notamment sur la page d'accueil à la place de l'article entier ou tronqué.

Cependant, contrairement à la fonction "more", l'utilisation de cette option n'est possible qu'avec le marqueur `php the_excerpt`. Pensez donc à bien modifier votre thème de la même manière que pour la fonction "more", pour utiliser l'extrait.

Insertion de vignettes dans votre thème…

Depuis quelques années, la notion de blog a beaucoup évolué. Le fameux journal des débuts a progressivement laissé place à des thèmes plus complexes, notamment avec les thèmes magazines.

Ce type de thème a apporté notamment une fonction intéressante qui consiste à afficher une image associée à un article sur la page d'accueil du site. Un exemple typique de ce genre de blog se trouve à la Figure 6.46.



Figure 6.46
Exemple de blog avec vignette associée à un article.

Dans cette partie, nous allons voir comment insérer ce type d'image dans votre thème. Au chapitre suivant, nous verrons comment le mettre en forme et l'intégrer visuellement dans le thème.

… *via* la fonction interne à WordPress

Depuis la version 2.9, WordPress a une fonction de base qui permet d'insérer des vignettes dans les articles. Cette fonction reste très simple à comprendre et à mettre en place.

Tout d'abord, il va falloir activer cette fonction en ajoutant une ligne de code dans le thème. Cette ligne de code est la suivante :

```
add_theme_support( 'post-thumbnails' );
```

Ensuite, vous allez insérer le code correspondant dans le thème pour pouvoir afficher cette vignette. Ce code est :

```
<?php the_post_thumbnail('thumbnail'); ?>
```

Nous allons placer ce code entre le titre et le contenu de l'article :

```
<h2><a href="<?php the_permalink(); ?>" title="<?php the_title(); ?>"><?php
the_title(); ?></a><div id="home-comments"><?php comments_popup_link(' (0)', '
(1)', ' (%)'); ?></div><?php edit_post_link('Editer', ' &#124; ', ''); ?></h2>
```

```
<?php the_post_thumbnail('thumbnail'); ?>

<div class="post_content">
  <?php the_content(); ?>
</div>
```

Pour vérifier que votre code fonctionne comme il faut, vous allez insérer une vignette dans un de vos articles. Comme nous l'avons déjà vu au Chapitre 3, pour insérer une vignette il vous faut cliquer sur le bloc Image à la une en bas à gauche de la page de rédaction d'un article (voir Figure 6.47).



Figure 6.47
Insérer une "image à la une".

Une fois la photo insérée, validez le tout et allez voir sur votre blog si tout fonctionne comme il faut. Votre image apparaît bien entre le titre et le contenu de l'article que vous venez de modifier.

Maintenant, il vous faut choisir une taille pour cette vignette. Pour ce faire, allez dans l'onglet Réglages, puis dans le sous-menu Médias.

On vous propose ici de donner une taille pour les miniatures à utiliser. Nous allons mettre 150 pixels (voir Figure 6.48).



Figure 6.48
Ajuster la taille des vignettes à 150 pixels.

Validez le tout. Le problème avec cette nouvelle fonction est qu'elle ne permet pas encore de régénérer les vignettes déjà créées pour le site. Ici, on a déjà généré des vignettes de tailles différentes, et WordPress ne sait pas régénérer une nouvelle vignette avec les nouvelles dimensions affichées.

Plusieurs possibilités se présentent alors : vous chargez une nouvelle fois la photo dans votre article, en supprimant au préalable la précédente ; mais vous pouvez très bien utiliser une extension, Regenerate Thumbnails (**http://www.viper007bond.com/wordpress-plugins/regenerate-thumbnails/**), qui va régénérer toutes les vignettes du site dans les nouvelles tailles. Cette seconde alternative devrait être disponible par défaut dans une prochaine version de WordPress.

Une fois les modifications faites, retournez sur votre blog, et vous pouvez voir que la vignette a changé de taille et fait maintenant 150 pixels.

Maintenant, peut-être souhaitez-vous également avoir cette image sur la page de votre article ? Rien de plus simple pour cela, vous allez insérer la même fonction the_post_thumbnail mais en lui donnant cette fois-ci un attribut pour que WordPress utilise non plus la "taille des miniatures" des images que vous importez, mais la "taille moyenne" (onglet Réglages, sous-menu Médias).

Le code à insérer dans le fichier single.php sera le suivant :

```
<?php the_post_thumbnail('medium'); ?>
```

Il sera à placer toujours entre le titre et le contenu de l'article.

… *via* l'utilisation des champs personnalisés

Cette utilisation, assez spécifique, n'est pas aussi flexible que celle présentée ci-dessus. Cela dit, elle est toujours d'actualité sur différents sites et, surtout, elle vous permettra de comprendre l'utilisation des champs personnalisés, qui sont une des fonctions puissantes de WordPress.

Tout comme l'extrait, les champs personnalisés sont une option avancée de la rédaction d'article sur WordPress (voir Figure 6.49).

Figure 6.49
Champs personnalisés.

Un champ personnalisé est composé de deux données :

- une clé ;
- une valeur.

Ces deux variables vont toujours de pair et ne fonctionnent qu'ensemble. La clé est le nom de la donnée méta. Elle figure dans le code du template et va permettre l'affichage de la valeur. Par exemple, si vous souhaitez afficher une image dans les articles de votre page d'accueil, la clé pourra s'appeler par exemple `vignette_home` et la valeur sera l'URL de l'image à afficher.

Comment les paramétrer ?

Pour utiliser les champs personnalisés sur votre blog, il va falloir les paramétrer dans votre thème WordPress. Le marqueur de base employé pour les champs personnalisés est :

```
< ?php the_meta(); ?>
```

Il faut impérativement que ce marqueur soit placé à l'intérieur de votre boucle WordPress pour être actif.

Pour créer des vignettes à l'aide des champs personnalisés, nous allons tout d'abord insérer la structure HTML qui nous permettra d'afficher une image. Cette structure est :

```
<a href= "URL du lien"><img src= "URL de l'image" /></a>
```

L'image sera associée à un lien qui pointera vers l'article en question.

Nous devons tout d'abord utiliser un marqueur WordPress nous permettant de récupérer l'URL de l'article automatiquement. Ce marqueur est :

```
<?php the_permalink();?>
```

Nous allons l'intégrer à la structure de base HTML. Nous obtenons alors :

```
<a href= "<? php the_permalink(); ?>" > <img src= "URL de l'image" /></a>
```

Maintenant, il va falloir insérer l'URL de l'image. Nous allons faire appel à un marqueur spécifique aux champs personnalisés pour récupérer l'URL de nos images. Nous n'utiliserons pas le marqueur présenté plus haut pour l'insertion des images, mais un autre, plus complet et plus approprié à ce genre d'usage. Ce marqueur est :

```
< ?php get_post_custom_values($key) ; ?>
```

Ce marqueur va nous permettre de paramétrer la clé et donc d'utiliser les champs personnalisés de manière très souple. Ici, nous pourrons employer autant de marqueurs que nous voudrons avec chacun une clé différente.

Dans notre exemple, nous appelons cette clé `vignette_home` pour obtenir le code suivant :

```
< ?php get_post_custom_values(" vignette_home ") ;?>
```

Cependant, ce code ne va pas suffire pour récupérer la valeur du champ personnalisé. Il faut justement lui ajouter la notion de "valeur" :

```
< ?php $values = get_post_custom_values(" vignette_home") ; echo $values[
0] ; ?> " id= " vignette_home " />
```

Ici, WordPress ira chercher et affichera toutes les valeurs ayant comme clé `vignette_home`.

Ensuite, pour ce qui concerne l'insertion de l'URL de l'image, nous pourrions très bien la saisir entièrement dans le champ personnalisé, mais ce serait un peu long. Nous pourrions également saisir directement l'URL du blog dans le code à insérer plutôt que d'utiliser un marqueur WordPress. Mais cette solution ne serait pas optimale car elle ne serait pas réutilisable sur un autre domaine.

Nous allons employer un marqueur WordPress pour récupérer l'URL du blog. Ainsi, vous pourrez utiliser votre thème sur un autre blog et donc sur un autre nom de domaine tout en gardant cette fonctionnalité. La partie à saisir dans le champ personnalisé sera l'URL de l'image à partir du dossier wp_content.

Le marqueur que nous allons utiliser pour récupérer l'URL du blog est :

```
<?php echo get_option('Home'); ?>
```

Avec ce marqueur et le morceau de code précédent, nous avons l'URL complète de notre image. Il va nous falloir maintenant l'intégrer dans la structure HTML de l'image :

```
<img src= ""< ?php echo get_option('Home'); ?><? php $values = get_post_custom_
values("Image"); echo $values[ 0]; ?>" id= "vignette_home" />
```

En intégrant ce code dans le reste de la fonction, nous obtenons la structure globale suivante :

```
<a href="<? php the_permalink(); ?>"> <img src= "<? php echo get_option('Home');
?><?php $values = get_post_custom_values("vignette_home"); echo $values[ 0];
?>" id= "vignette_home" /></a>
```

Comme nous souhaitons utiliser cette fonction au niveau de la page d'accueil, nous devons insérer ce code dans le fichier index.php, et plus précisément dans la boucle WordPress. Vous allez donc ouvrir ce fichier avec votre éditeur de texte et y insérer le code pour afficher les vignettes à l'endroit où vous voulez les voir apparaître sur votre page web.

Enregistrez les modifications et rédigez un article pour tester la fonctionnalité. Au niveau des champs personnalisés, vous allez entrer les informations suivantes (voir Figure 6.50) :

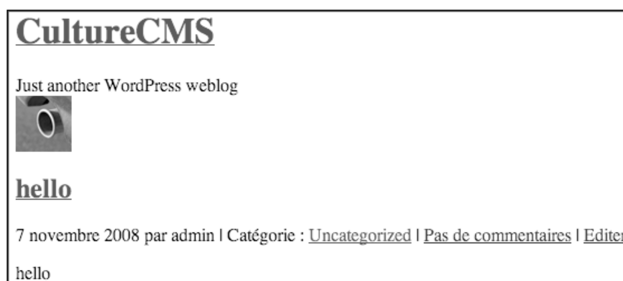
- Clé : vignette_home.
- Valeur : wp-content/uploads/2008/11/image.jpg (insérez ici l'URL de l'image à partir de wp-content).

Figure 6.50
Insertion d'un champ personnalisé.

Enregistrez le champ créé et publiez votre article.

Rendez-vous sur la page d'accueil de votre blog ; l'image que vous avez ajoutée dans le champ personnalisé apparaît maintenant à côté de votre article, en fonction de l'endroit où vous avez inséré votre code (voir Figure 6.51).

Figure 6.51
Insertion d'une vignette sur la page d'accueil.



À vous ensuite de positionner et de dimensionner l'image en fonction de vos besoins au moment de la création du style de votre blog. Vous pouvez décider d'avoir une petite vignette à gauche du contenu, ou une plus grosse au-dessus du titre des articles. Voici quelques exemples qui vous aideront à mieux choisir (voir Figures 6.52 à 6.54).

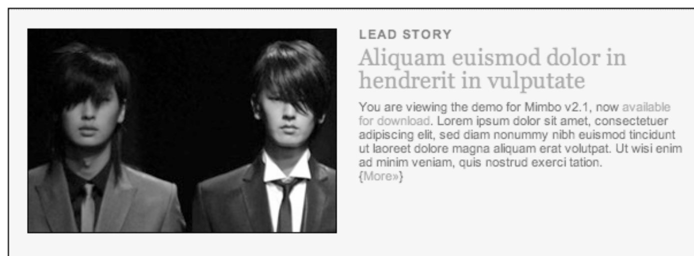


Figure 6.52
Affichage de vignette 1.

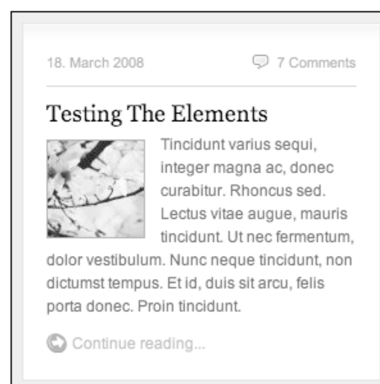


Figure 6.53
Affichage de vignette 2.



Figure 6.54
Affichage de vignette 3.

Création d'un modèle de page

Un modèle de page permet de choisir le contenu que vous allez voir apparaître sur vos pages. Par exemple, vous pouvez décider d'afficher la colonne latérale sur une page du blog mais pas sur une autre. Vous utiliserez alors deux modèles de page différents.

Leur conception est comparable à celle d'un fichier du thème. Vous allez y reprendre l'ensemble des parties du blog et y insérer le contenu que vous voulez voir apparaître. Voici par exemple le contenu du modèle de page qui affichera la blogoliste pour le thème default :

```
<?php
/*
Template Name: Liens
*/
?>
<?php get_header(); ?>
<div id="content" class="widecolumn">
  <h2>Liens :</h2>
  <ul>
    <?php get_links_list(); ?>
  </ul>
</div>
<?php get_footer(); ?>
```

Tout d'abord, vous avez la structure suivante :

```
<?php
/*
Template Name: Nom du modèle de page
*/
?>
```

Vous allez ici insérer le nom que vous donnez à ce modèle de page. Il vous permettra de distinguer les différents modèles au moment d'en choisir un pour une de vos pages.

Ensuite, vous retrouvez l'en-tête et le marqueur qui va vous permettre d'afficher la blogoliste :

```
<?php get_links_list(); ?>
```

Et enfin, vous insérez le pied de page. Notez que ce modèle de page n'affiche pas de colonne latérale.

Ce template s'appelle links.php (voir Figure 6.55) et est situé au même niveau que les autres fichiers du blog.

Si vous souhaitez créer un modèle de page spécifique, vous devez reprendre le même raisonnement que pour ce modèle-ci. Vous pourrez insérer tous les éléments que vous voulez voir s'afficher sur une nouvelle page de votre blog.

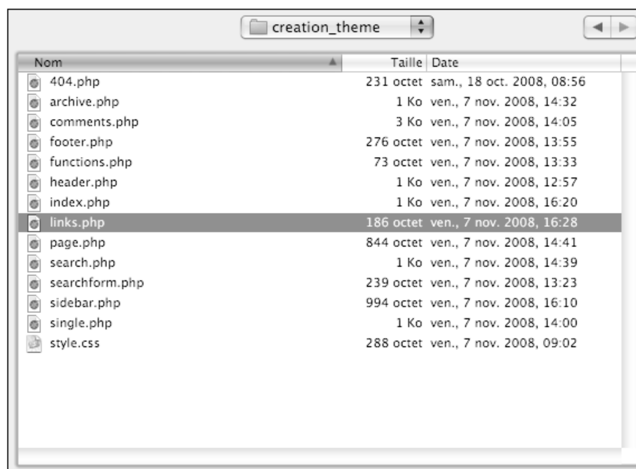


Figure 6.55
Fichier links.php.

Lorsque le modèle sera enregistré, vous pourrez créer votre nouvelle page et le choisir parmi les modèles qui vous seront proposés (voir Figure 6.56).

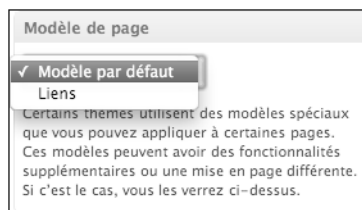


Figure 6.56
Affichage des différents modèles de page.

Création d'un thème enfant

WordPress permet la création de déclinaisons pour les thèmes. C'est ce qu'on appelle un thème enfant. En fait, vous créez un thème en fonction d'un autre thème qui, lui, sera appelé thème parent.

Prenons par exemple le thème default. Vous souhaitez le modifier pour lui donner une apparence plus personnelle. Plutôt que de modifier directement les fichiers de thème, vous allez créer un thème enfant du thème default et lui apporter vos propres changements.

Pour ce faire, vous allez créer un nouveau dossier, avec le nom de votre thème enfant. Dans ce dossier, vous créerez une nouvelle feuille de style. Comme pour la conception d'un thème classique, vous insérerez les informations qui permettront de différencier ce thème des autres :

```
/*
Theme Name: thème enfant WordPress
Theme URI: http://www.fran6art.com/
Description: Thème créé de A à Z avec le livre Campus WordPress.
Version: 1.0
Author: Francis Chouquet
```

```
Template: default
Author URI: http://www.fran6art.com/

Creation Theme by Francis Chouquet || http://www.fran6art.com
*/
```

Ici, les informations sont les mêmes que pour n'importe quel thème, si ce n'est que nous avons ajouté une ligne `template: default` qui va permettre à WordPress d'associer ce thème au thème parent default.

Ainsi, si vous refermez maintenant la feuille de style, que vous transférez le dossier de ce thème sur votre hébergement, sous le dossier wp-content > themes, et que vous allez sur la page de gestion des thèmes sur l'interface d'administration, vous verrez ce nouveau thème apparaître. Activez-le et vous constaterez qu'il utilise les fichiers du thème default. WordPress, en lisant les informations de la feuille de style, a compris que c'était un thème enfant de default.

Maintenant, si vous rouvrez la feuille de style de votre thème enfant, vous allez pouvoir y faire toutes les modifications que vous voulez. Celles-ci seront prises en compte tout en continuant d'utiliser les fichiers du thème default. Et depuis la version 2.7 de WordPress, vous avez la possibilité de faire de même avec tous les fichiers et pas uniquement avec la feuille de style.

Admettons que vous souhaitiez changer l'en-tête du blog. Vous pouvez créer un nouveau fichier header.php, que vous allez intégrer dans le dossier de votre thème enfant. Au moment de l'affichage du thème, WordPress verra qu'il y a un fichier header.php et l'utilisera à la place du fichier header.php du thème parent.

L'un des principaux avantages des thèmes enfants est de pouvoir modifier des thèmes sans avoir besoin de corriger les fichiers. À l'instar des extensions, les thèmes sont souvent mis à jour. Si vous faites des modifications dans un thème existant, il vous sera impossible de le mettre à jour par la suite. Tandis qu'en utilisant un thème enfant cette mise à jour ne posera aucun problème.

7 Concevoir le design de son blog

Si le chapitre précédent demandait quelques notions en PHP, ce chapitre va exiger que vous maîtrisiez quelques connaissances de base en CSS (*Cascading Style Sheets*).

Commencer avec un croquis

La réalisation d'un design de site web, et notamment ici d'un blog, se fait par étapes. En premier lieu, vous devez mettre en relief vos idées.

Vous avez forcément des idées pour votre blog, et plutôt que de passer directement sur un logiciel, il est important de prendre le temps de coucher ces idées sur le papier. Pas besoin ici d'être un excellent dessinateur, vous allez uniquement essayer de faire un rapide croquis de votre blog, avec les différents emplacements, voire des indications comme "tons clairs/ tons sombres", ou même des tailles en pixels si vous souhaitez obtenir un croquis précis.

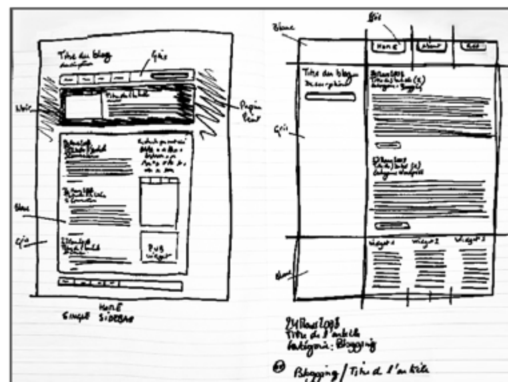
Pour notre exemple, vous allez réaliser un thème simple à partir des prérequis suivants :

- colonne principale large, pour laisser la place au contenu ;
- colonne latérale à droite ;
- barre de navigation horizontale, qui laisse apparaître les sous-catégories au survol ;
- fond original en dégradé ;
- police Serif, telle que Georgia par exemple ;
- titre du blog écrit avec une police originale et placé dans une image.

Avec ces informations, vous pouvez établir un croquis qui donne déjà un aperçu simple, mais que vous pouvez facilement faire évoluer. C'est le principe même du croquis : donner une première idée en quelques minutes, idée que l'on peut faire évoluer ou reprendre autant de fois que l'on veut.

Le croquis sur lequel nous allons baser notre feuille de style est représenté à la Figure 7.01.

Figure 7.01
Croquis de départ.



Placement des différents éléments du blog

Maintenant que vous avez la structure globale du thème, vous allez positionner les principaux éléments que sont :

- l'en-tête ;
- le contenu ;
- la colonne latérale ;
- le pied de page.

Éléments généraux du thème

Tout d'abord, vous allez mettre les premières règles du style pour la balise `body` du code HTML. Cette balise contient l'ensemble du blog, et ces règles seront donc applicables à tous les éléments HTML du blog. Cependant, ces règles ne sont pas figées dans le marbre, elles seront modifiables dès lors que vous donnerez une indication plus précise au niveau d'une balise plus locale.

Par exemple, vous allez définir une famille de polices pour l'ensemble du blog. Il se peut que par la suite vous décidiez d'utiliser un style de police différent pour les titres d'articles. Vous définirez donc ce style au niveau de la balise concernée. Ce style prendra le pas sur celui défini au départ.

Commencez par ouvrir votre feuille de style style.css et insérez le code suivant directement sous les informations du thème que vous avez intégrées plus tôt :

```
body {
  font-family: "Georgia", Times, Serif;
  font-size: 0.9em;
  text-align: left;
  background: #fff;
  color: #444;
  margin: 0;
  padding: 0;
}
```

Dans ce sélecteur pour la balise `body`, vous avez précisé la famille de polices que vous souhaitez utiliser. Ici, le choix s'est porté sur des polices Serif : Georgia en première option, puis Times. Si la police Georgia n'est pas installée sur l'ordinateur qui affiche la page web, Times sera le second choix. Et si celle-ci n'est pas non plus installée, alors ce sera la police de type Serif de base.

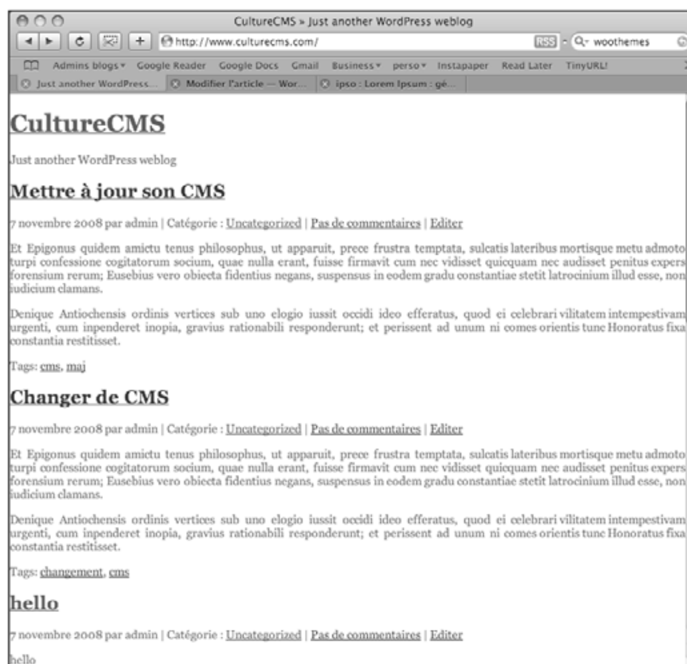
La taille du texte est pour le moment fixée à 0,9 em. L'avantage de l'unité "em" est qu'elle changera de taille en fonction du zoom effectué sur la page. Par chance, aujourd'hui, les derniers navigateurs prennent le changement de taille en compte, même sur des tailles en pixels.

Ensuite, vous avez positionné le texte à gauche, donné un fond blanc à l'ensemble du blog, puis vous avez choisi une couleur gris sombre pour le texte. Le noir reste la couleur la plus lisible, mais le gris sombre donne une certaine nuance au texte qui est plus esthétique.

Enfin, vous avez appliqué ce qu'on appelle un "reset CSS" très simple. Le but de ces deux dernières lignes est de dire dès le départ qu'il n'y a aucune marge intérieure (`padding`) ni extérieure (`margin`) pour aucun élément. En effet, certains navigateurs appliquent une feuille de style par défaut qui a tendance à biaiser la mise en forme d'un site web. Faire cette précision dès le départ évite les surprises par la suite pour les positionnements de certains blocs avec différents navigateurs.

Enregistrez votre feuille de style et ouvrez votre navigateur sur la page de votre thème. L'ensemble n'a pas encore beaucoup évolué, mais il affiche déjà le résultat des règles que vous venez d'insérer (voir Figure 7.02).

Figure 7.02
Insertion de la balise *body* dans le thème.



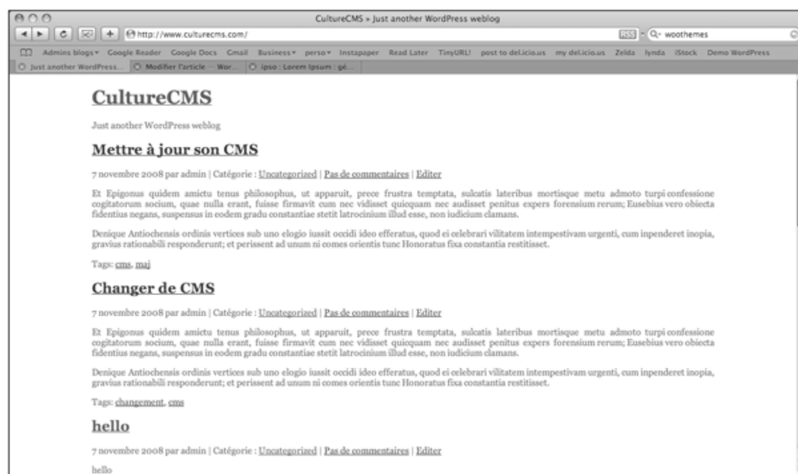
Vous allez maintenant attribuer quelques règles à la balise qui dispose de l'ID `page`. Cette balise englobe l'ensemble des différents éléments du blog. Elle ne comprend pas toute la page web, contrairement à la balise `body`, mais uniquement le contenu affiché, ce qui est une nuance subtile mais nécessaire.

Vous allez donc insérer les lignes suivantes, directement sous le style du sélecteur `body` :

```
#page {
  margin: 0 auto 0 auto;
  width: 960px;
}
```

Enregistrez les modifications et rafraîchissez votre navigateur : votre blog est désormais aligné au centre de la page web et possède une largeur de 960 pixels (voir Figure 7.03).

Figure 7.03
Alignement du thème au centre de la page web.



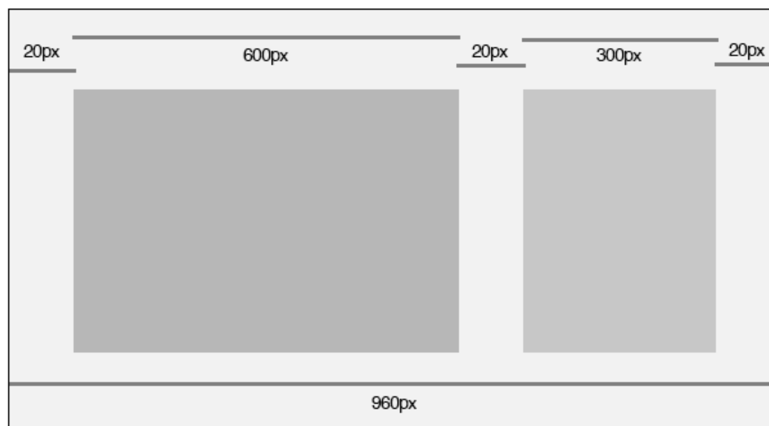
Cette largeur de 960 pixels est choisie de façon arbitraire. Vous pouvez très bien opter pour une autre largeur, en fonction de vos besoins.

Positionnement des différents éléments

Vous allez maintenant dimensionner et positionner chacun des éléments du blog, en commençant par donner une taille au contenu de la page et à la colonne latérale.

La largeur globale du contenu est de 960 pixels. Nous appliquerons une marge extérieure de 20 pixels de chaque côté et de 20 pixels à l'intérieur. Pour ce thème, nous aurons une colonne latérale de 300 pixels qui sera assez large pour accueillir toutes les informations nécessaires. Nous lui appliquerons également une marge extérieure de 20 pixels à droite et de 20 pixels à gauche (voir Figure 7.04).

Figure 7.04
Largeur et marges du thème.



Insérez le code suivant pour votre colonne latérale, en reprenant les informations données ci-dessus :

```
/* sidebar */

.sidebar {
  width: 300px;
  margin: 0 20px;
}
```

Notez que toutes les parties du fichier style.css sont commentées pour pouvoir les délimiter. Ces commentaires sont une nouvelle fois en anglais, dans le cas où vous souhaiteriez par la suite proposer votre thème au téléchargement.

Enregistrez votre fichier style.css.

Vous allez maintenant définir une taille pour le contenu de la colonne principale, qui sera positionnée sur la gauche. Vous savez que cette colonne fait 960 pixels de large, que la colonne latérale fait 300 pixels de large, avec une marge à droite de 20 pixels et une marge à gauche de 20 pixels. N'oubliez pas non plus que le contenu doit également avoir une marge à gauche de 20 pixels. À partir de tout cela, il vous faut calculer la taille à respecter pour que les blocs tiennent les uns dans les autres. Ce calcul est une simple soustraction à partir des largeurs déjà définies :

960 – 300 – 20 – 20 – 20 (appliqués à gauche du contenu) = 600 pixels.

Vous insérerez le code suivant au-dessus des informations de la colonne latérale pour garder une cohérence par rapport à la structure même du blog :

```
/* home content */

#content {
  width: 600px;
  margin-left: 20px;
}
```

Enregistrez les modifications et rafraîchissez une nouvelle fois votre navigateur : votre contenu est maintenant compris dans un bloc de 620 pixels, avec une marge extérieure à gauche de 20 px (voir Figure 7.05).

Vous pouvez maintenant positionner ces deux colonnes l'une à côté de l'autre. Pour cela, vous utiliserez l'élément `float` pour placer la colonne principale à gauche et la colonne latérale à droite. Dans le code de chacun des blocs, vous allez ajouter le code suivant.

À placer dans le sélecteur d'ID `content` :

```
float: left;
```

À placer dans le sélecteur de classe `sidebar` :

```
float: right;
```

Ce qui donne comme code pour ces deux éléments :

```
/* home content */

#content {
```

```
  width: 600px;
  margin-left: 20px;
  float: left;
}

/* sidebar */

.sidebar {
  width: 300px;
  margin: 0 20px 0 20px;
  float: right;
}
```

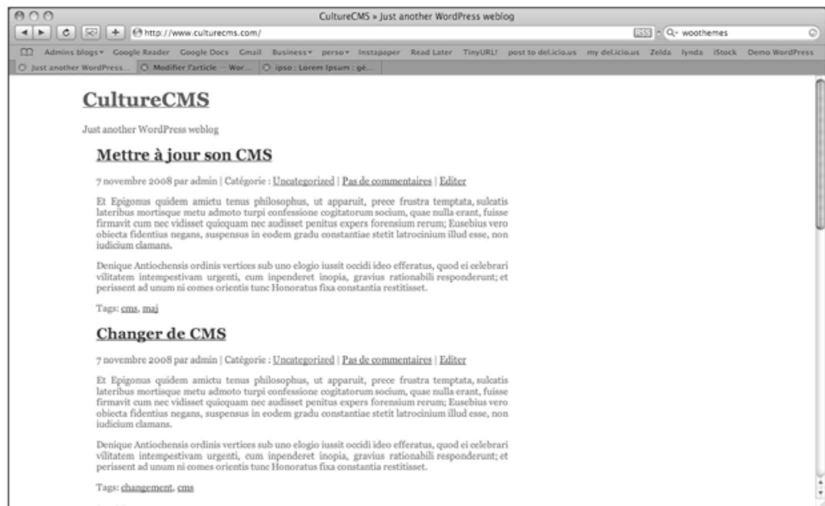


Figure 7.05

Affichage du contenu.

Enregistrez les modifications et allez voir le résultat dans le navigateur : les deux colonnes sont désormais côte à côte (voir Figure 7.06).

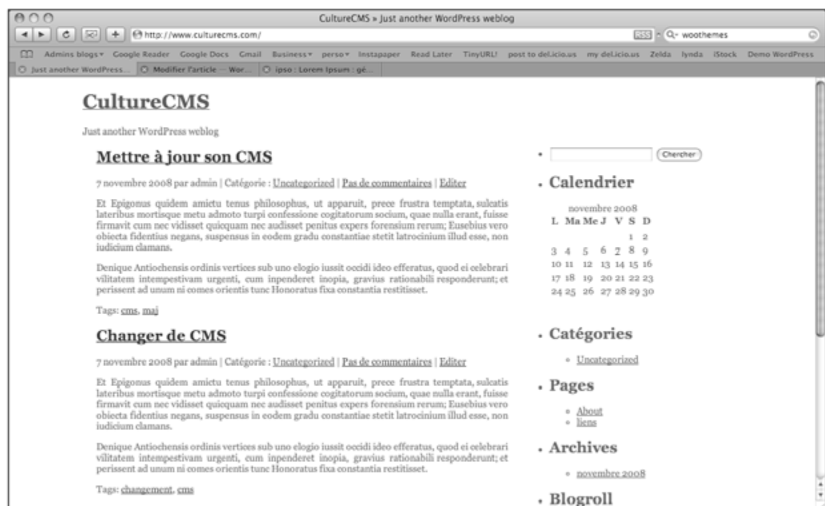


Figure 7.06

Alignement des colonnes.

Cependant, le pied de page (footer) est encore mal positionné, puisqu'il se trouve sous la colonne latérale. Vous allez devoir le placer sous le contenu (la colonne principale) et pour cela utiliser la règle `clear: both`. La règle `clear` annule les positionnements flottants en cours, ce qui vous permet de remettre le pied de page à la suite des deux blocs qui le précèdent. Vous allez en profiter ici pour donner au pied de page les bonnes marges extérieures, à savoir 20 pixels à gauche et à droite.

Insérez le code suivant directement sous le bloc `sidebar` :

```
/* footer */

#footer {
  clear: both;
  margin: 0 20px;
}
```

Enregistrez le tout et rafraîchissez le navigateur : le pied de page est maintenant positionné sous le contenu et la colonne latérale.

Tous les éléments sont à présent bien positionnés. Mais pour garder une cohérence par rapport à l'alignement vertical de l'ensemble du blog, vous devez encore définir les marges extérieures gauche et droite pour l'en-tête. Si vous regardez sur votre navigateur, ces deux éléments sont décalés de 10 pixels par rapport au contenu (voir Figure 7.07).

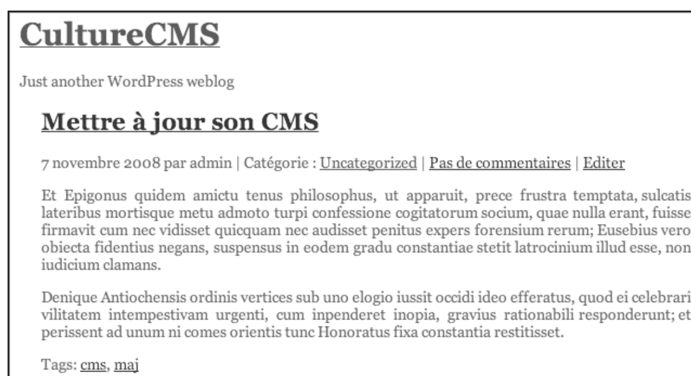


Figure 7.07
Décalage de l'en-tête.

Vous allez donc créer une ligne pour personnaliser l'en-tête de votre blog, ligne que vous placerez au-dessus du commentaire `/* home content */` et qui contient le style suivant :

```
/* header style */

#header {
  margin: 0 10px;
}
```

Enregistrez la feuille de style et rechargez la page dans le navigateur : tous les éléments sont maintenant alignés les uns par rapport aux autres (voir Figure 7.08). Vous avez terminé la première étape de la personnalisation du thème.

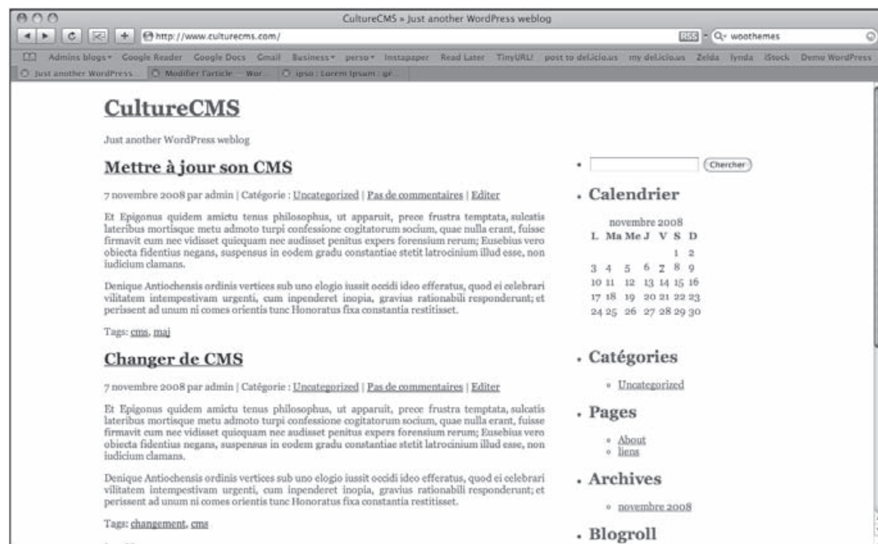


Figure 7.08

Alignement des différents éléments.

Mise en place du design : l'en-tête et le fond du blog

Dans cette section, vous allez placer les différents éléments graphiques de votre thème. Comme vous l'avez vu au début de ce chapitre, vous allez intégrer un fond tout autour du blog. Ici, ce sera une image avec un motif qui se répète. Vous ajouterez également une image pour l'en-tête. Tous les fichiers pour réaliser ces étapes sont fournis sur **http://apprendre-wordpress.fr/livre**.

Mais, tout d'abord, vous commencerez par signaler explicitement la bordure du blog d'un trait noir épais. Pour cela, vous allez insérer la ligne de code suivante dans le sélecteur `page` :

```
border: 5px solid #999;
```

Enregistrez votre travail et rafraîchissez votre navigateur : vous avez bien maintenant un cadre qui entoure votre contenu (voir Figure 7.09).

Vous allez à présent placer une image de fond pour toute la partie extérieure au blog, qui s'intégrera comme des tuiles qui se suivent. L'image à utiliser se nomme "background.jpg" sur **http://apprendre-wordpress.fr/livre**. Vous allez la placer dans un nouveau dossier, créé au niveau des fichiers du thème, que vous nommerez "images". Vous pointerez alors vers elle depuis le sélecteur de la balise `body` grâce à la règle CSS suivante :

```
background: #555 url(images/background.jpg);
```

Si vous allez sur votre navigateur une fois que cette règle est mise en place, vous constatez que le fond vient se placer derrière l'ensemble de la page, y compris le contenu textuel de vos articles ou même la colonne latérale (voir Figure 7.10).

Figure 7.09
Bordure du thème.

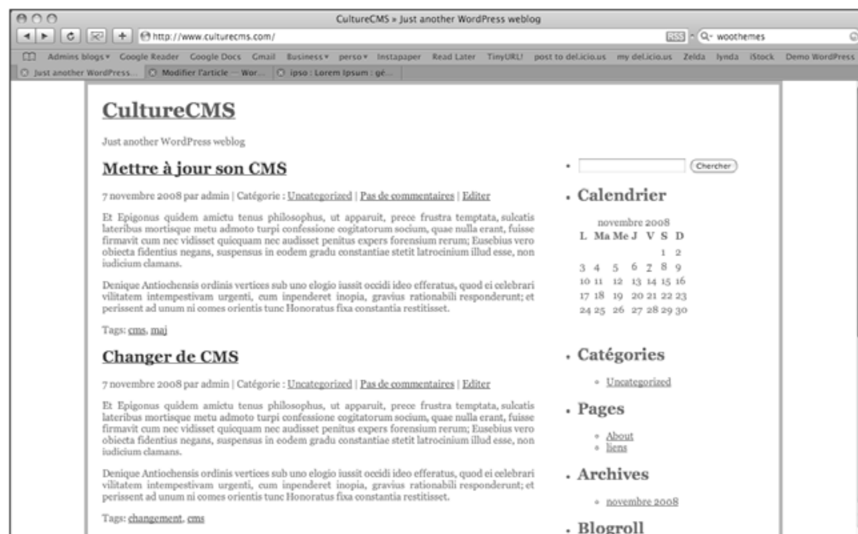
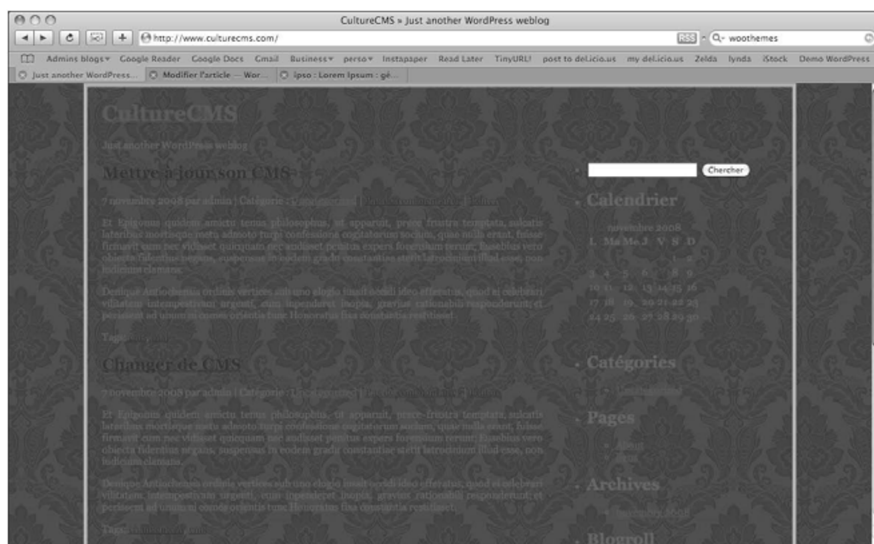


Figure 7.10
Affichage du fond sur l'ensemble du blog.



Ceci est dû au fait que vous n'avez pas encore établi de couleur de fond pour le sélecteur `page`, et donc que rien ne vient cacher l'image de fond (voir Figure 7.11). Choisissez le blanc pour cette couleur de fond :

```
background-color: white;
```

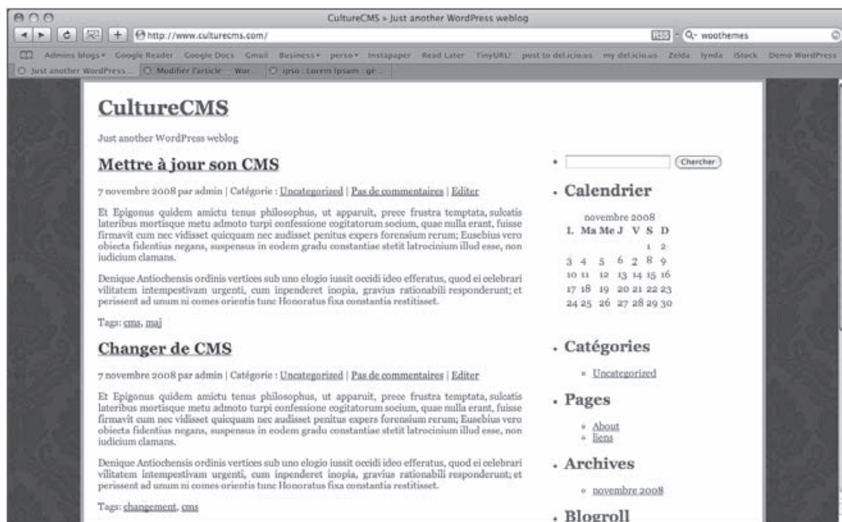


Figure 7.11
Affichage du fond après le paramétrage de la balise *page*.

Vous allez par ailleurs en profiter pour décaler le cadre du haut de la page web pour laisser apparaître un morceau du fond et ainsi mettre encore un peu plus en relief votre contenu. Vous ferez de même pour le bas de la page. Pour réaliser ces marges, vous allez modifier une partie des attributs du sélecteur `page`. Remplacez la ligne suivante :

```
margin: 0px auto 0px auto;
```

par

```
margin: 30px auto 30px auto;
```

Ici, vous venez de placer des marges extérieures de 30 pixels en haut et en bas de votre blog. L'ensemble du code pour le sélecteur `page` doit maintenant ressembler à cela :

```
#page {
  margin: 30px auto;
  width: 960px;
  border: 5px solid #999;
  background-color: white;
}
```

Vous allez ensuite placer une image sur le blog, qui a son importance puisqu'il s'agit de la bannière pour l'en-tête. Cette image, dont le fichier est header_bg.png (disponible sur **http://apprendre-wordpress.fr/livre**), a pour dimensions 930 x 212 pixels. Vous allez donc positionner cette image également dans le dossier images du thème.

Pour accueillir l'ensemble de l'image sans casser votre design, vous devez agrandir l'en-tête. Mais auparavant, vous allez l'insérer telle quelle pour voir pourquoi il va falloir l'agrandir.

C'est à partir du sélecteur `header` que vous allez insérer l'image, *via* le code suivant :

```
background-image: url(images/header_bg.png);
```

Comme vous le constatez en rechargeant la page, l'image ne tient pas dans l'en-tête. Vous devez agrandir ce dernier pour laisser apparaître l'ensemble de l'image. La hauteur sera de 212 pixels, c'est-à-dire la taille de l'image. Pour cela, ajoutez le code suivant, toujours dans le sélecteur `header` :

```
height: 213px;
```

Vous allez également en profiter pour centrer un peu mieux le tout en donnant des marges supplémentaires à l'ensemble. Vous remplacerez donc le code suivant dans la balise `header` :

```
margin: 0px 10px 0px 10px;
```

par

```
margin: 10px 20px 10px 20px;
```

Le contenu du sélecteur `header` doit maintenant ressembler à ceci :

```
/* header style */

#header {
  margin: 10px 20px 10px 20px;
  background-image: url(images/header_bg.png);
  height: 213px;
}
```

Rechargez la page : votre blog commence à prendre forme ! Le fond est placé correctement ainsi que l'image de l'en-tête. Cependant, il faudrait faire disparaître le titre et la description du blog puisque nous allons les remplacer par l'image de l'en-tête.

Pour cacher ces informations, vous allez juste les rendre invisibles en les déplaçant très loin sur la gauche. Notez bien qu'il n'est pas question ici de supprimer le titre et la description de l'en-tête, car vous rendriez leur contenu invisible pour les personnes malvoyantes comme pour Google – l'accessibilité profite à tous…

Le titre du blog s'affiche avec la balise `h1`. C'est un titre cliquable ; vous allez donc le rendre invisible en ciblant la combinaison de sélecteurs `h1 a` avec CSS. Pour la description, vous rendrez invisible la balise paragraphe (p) de l'en-tête, donc la combinaison de sélecteurs est `#header p`.

Le code à insérer est donc :

```
h1 a {
  position: absolute;
  left: -5000px;
  top: -5000px;
  text-indent: -5000px;
}

#header p {
  position: absolute;
  left: -5000px;
  top: -5000px;
  text-indent: -5000px;
}
```

Vous placerez ce code directement sous la balise `header`. Ici, vous donnez une position absolue à ces deux éléments et ils seront donc constamment placés à 5 000 pixels sur la gauche de l'écran, donc invisibles.

Enregistrez les modifications et rafraîchissez votre navigateur : le titre et la description du blog ont disparu, vous ne voyez plus que l'image d'en-tête (voir Figure 7.12).



Figure 7.12
Affichage de l'image d'en-tête.

Mise en place du design : le menu de navigation

Sur le croquis, vous aviez émis la volonté d'avoir un menu horizontal de navigation, avec des sous-catégories en format "menu déroulant" qui apparaissent au passage de la souris (drop down menu). C'est ce que vous allez mettre en place ici.

Pour cela, vous commencerez par insérer le code du menu dans le fichier header.php. En effet, vous allez placer la barre de navigation directement sous l'image de l'en-tête, donc sous la balise avec l'ID `header`. Nous avons choisi d'insérer le code ici en partant du principe que le menu est encore dans la partie en-tête du blog.

Ouvrez votre fichier header.php et ajoutez le code suivant tout en bas du fichier, après toutes les données déjà insérées :

```
<div id="navbar">
<ul id="nav2"><?php wp_nav_menu(); ?></ul>
<?php get_search_form(); ?>
</div>
```

Ce menu de navigation est divisé en deux parties :

- Le menu de navigation lui-même, avec les différentes catégories du blog.
- Un formulaire de recherche, collé sur la droite. Nous avons déjà un formulaire de recherche dans la colonne latérale, mais il peut être intéressant de l'avoir directement dans le menu. À vous de choisir l'endroit où vous souhaitez le garder.

Dans le code, on reprend donc tout simplement la nouvelle fonction `<?php wp_nav_menu(); ?>` apparue avec WordPress 3 et qui vous permettra, comme nous l'avons vu au Chapitre 4, de créer un menu totalement customisable. On y ajoute également la ligne pour le formulaire de recherche. Mais, pour que cette gestion du menu soit opérationnelle, il faut créer une nouvelle ligne dans le fichier functions.php que vous aviez créé au moment de la widgetisation de la sidebar. Ce morceau de code est le suivant :

```
add_theme_support( 'menus' );
```

À partir de là, vous pouvez utiliser le sous-menu Menus, sous Apparence, et créer le menu que vous souhaitez.

Rechargez votre blog, vous voyez la liste des catégories s'afficher (voir Figure 7.13).



Figure 7.13
Affichage des catégories et du formulaire de recherche dans le menu.

Vous allez maintenant devoir mettre tout ça en forme, car ce n'est pas très bien intégré ni très esthétique. Pour cela, vous insérerez les lignes de code CSS suivantes directement à la fin de votre fichier style.css :

```
/* navigation */

#navbar {
  display: block;
  float: left;
  background-color: #555;
  margin-left: 20px;
  margin-top: 10px;
  margin-right: 20px;
  width: 920px;
}
```

Vous commencez ici par donner les informations à l'ensemble du menu, c'est-à-dire à la balise avec l'ID "navbar". Tout d'abord, vous allez l'afficher en mode "block" et la faire

flotter à gauche. Ces informations sont très importantes pour le positionnement de l'ensemble, notamment la valeur "block" pour la déclaration "display" qui transforme votre liste verticale en liste horizontale.

Puis vous lui appliquez un fond gris sombre (#555) et ajoutez des marges de 20 pixels à gauche et à droite, ainsi qu'une marge extérieure supérieure de 10 pixels. Enfin, vous n'oubliez pas de donner une taille à ce menu : ce sera 920 pixels, ce qui équivaut à la largeur totale de 960 pixels moins les deux marges extérieures de 20 pixels chacune :

```
#nav {
  font-size: 13px;
  background: #555;
}
```

Ensuite, vous spécifiez quelques règles pour le sélecteur `#nav`, donc le menu des catégories. Vous définissez une taille de police de 13 pixels, ainsi que le même fond gris sombre `#555` :

```
#nav, #nav ul {
  list-style: none;
  float: left;
  line-height: 1.5;
  padding: 0;
  margin: 0;
  width: 655px;
}
```

Dans le code ci-dessus, vous avez inséré des règles qui s'appliquent à la fois à la balise contenant l'ID `nav` et à la balise `ul` qui se trouve à l'intérieur de cette balise, avec le sélecteur `#nav ul`. Dans une liste, par défaut, il y a un style avec des puces attribuées à chaque début de ligne.

En insérant le style `list-style: none;`, vous supprimez ces puces, qui auraient été peu esthétiques dans votre menu horizontal.

Il faut que les catégories apparaissent à gauche : vous donnez donc à l'ensemble un flottement à gauche. Ensuite, vous positionnez vos éléments HTML en attribuant une hauteur de ligne de 1,5 em et en enlevant toute marge extérieure et intérieure qui pourrait être héritée d'un élément supérieur (ou même de la feuille de style par défaut du navigateur).

Enfin, vous établissez une largeur pour ce menu, afin que l'ajout de catégories n'aille pas se superposer avec le formulaire de recherche.

Tout cela est très bien, mais il faut encore gérer l'activation du menu lors de son survol par la souris. Voilà le code à ajouter :

```
#nav a ,#nav a:hover{
  display: block;
  text-decoration: none;
  border: none;
  background-color: #555;
}
```

Ici, vous définissez un style pour les liens du menu à l'affichage et au survol (`:hover` étant une pseudo-classe CSS qui permet de gérer l'état où l'élément est survolé par la souris).

Tout d'abord, chaque lien s'affiche en `block`, ce qui permet de bien positionner chacun des éléments. Ensuite, vous retirez tout élément de style des liens – ce qui correspond à leur soulignement – qui n'est pas très esthétique dans un menu de navigation. Vous supprimez toute bordure qui pourrait être héritée, et enfin vous définissez la couleur de fond `#555`, qui va créer une harmonie avec le reste du style du menu.

Vous avez défini le style global du menu, mais il est possible aussi de donner un style aux lignes du menu :

```
#nav li {
  float: left;
  list-style: none;
  border-right: 1px solid #777;
}
```

Avec ce code, chaque ligne est positionnée et alignée à gauche, sans style de liste (donc pas de puce précédant chaque ligne), et vous séparez chacune de ces lignes avec une bordure grise de 1 pixel.

```
#nav a, #nav a:visited {
  display: block;
  color: #f5f5f4;
  padding: 6px 10px 6px 10px;
}
```

Précédemment, vous avez défini les règles qui s'appliquaient aux liens normaux et en état de survol. Avec le code ci-dessus, vous définissez les mêmes règles pour les liens et les liens visités (c'est-à-dire déjà cliqués par l'utilisateur), mais avec des valeurs un peu différentes. Vous retrouvez donc le positionnement en "block", mais la couleur grise est plus claire (`#f5f4f4`), et vous positionnez le tout pour que le texte soit bien centré sur la ligne :

```
#nav a:hover, #nav a:active {
  background: #000;
  text-decoration: none;
}
```

Enfin, vous déterminez ici le style commun pour les liens en survol et les catégories actives (c'est-à-dire l'état où l'utilisateur clique sur le lien). Dans le contexte défini par ces pseudo-classes, vous choisissez un fond noir pour mettre le lien ou la catégorie en relief, et vous retirez toute décoration du texte, telle que le soulignement.

Il reste à insérer le code concernant les menus déroulants, que l'on appelle menus "dropdowns". Ceux-ci apparaissent lorsque des sous-catégories existent, ce qui dans le code HTML est géré par PHP. Le code CSS à insérer dans style.css est le suivant :

```
/* Dropdown Menu */
#nav li ul {
  position: absolute;
  left: -999em;
  height: auto;
  width: 152px;
  border-bottom: 1px solid #777;
}
```

Décryptons ces lignes. Tout d'abord, le sélecteur CSS indique que le bloc de règles s'applique à la liste qui figure sous une ligne du menu ciblé par la classe `nav`. Dans votre menu déroulant, une ligne équivaut à une catégorie, et vous allez afficher une liste des sous-catégories (si elles existent).

En premier lieu, vous cachez le menu déroulant lorsque vous ne le survolez pas. Vous utilisez un positionnement fixe à 999 em à gauche et une hauteur établie de manière automatique.

Vous définissez ensuite une largeur de 152 pixels et attribuez à chaque ligne de sous-catégorie une bordure inférieure de 1 pixel et de couleur grise `#777`, bordure qui séparera les lignes entre elles.

Quelques ajouts esthétiques pour chaque ligne du sous-menu :

```
#nav li li {
  width: 150px;
  border-top: 1px solid #777;
  border-right: 1px solid #777;
  border-left: 1px solid #777;
  background: #777;
}
```

Vous spécifiez ici qu'elles feront 50 pixels de large, auront une bordure tout autour sauf en bas (puisque chaque bordure supérieure d'une ligne correspond à la bordure inférieure de la précédente, et la dernière bordure inférieure est déjà définie par la liste entière), et que leur fond sera de couleur grise (`#777`).

Vient ensuite un gros morceau de code CSS, qui va vous servir à gérer les liens des catégories, donc chaque ligne du sous-menu, dans les différents états nécessaires : normal, déjà visité, en cours de survol et en cours de clic.

```
#nav li li a, #nav li li a:visited{
  font-weight:normal;
  font-size:0.9em;
  color:#FFF;
}

#nav li li a:hover, #nav li li a:active{
  background:#000;
}

#nav li:hover ul, #nav li li:hover ul, #nav li li li:hover ul, {
left: auto;
}

a.main:hover{
  background:none;
}
```

Ces différentes lignes définissent le style pour les liens de vos sous-catégories. La taille de la police est de 0,9 pixel, et sa couleur est blanche. Quand vous survolez une de ces sous-catégories, le fond est noir et vous définissez le style pour avoir l'effet "dropdown" sur vos sous-catégories. Le `left: auto` permet de ramener la liste déroulante et de la faire apparaître au survol.

Ici, vous utilisez la pseudo-classe `:hover` pour faire réapparaître les menus déroulants que vous avez placés préalablement à 999 em à gauche de la page web.

Une fois le menu des catégories positionné et réglé avec le style qui convient, vous devez vous occuper de placer le formulaire de recherche. Il sera aligné à droite, sans marge extérieure et avec une marge intérieure de 4 pixels pour le haut et le bas, et de 10 pixels pour la gauche et la droite :

```
#navbar #searchform {
  float: right;
  text-align: right;
  margin: 0;
  padding: 4px 10px;
}
```

Lorsque vous avez inséré toutes les lignes de code que nous venons de voir, enregistrez votre fichier style.css et rafraîchissez une nouvelle fois la fenêtre de votre navigateur. Désormais, vous disposez d'un menu de navigation qui apparaît entre la bannière du blog et le contenu (voir Figure 7.14). Ce menu comprend un accès à la page d'accueil, les catégories et sous-catégories du blog, ainsi que le formulaire de recherche.

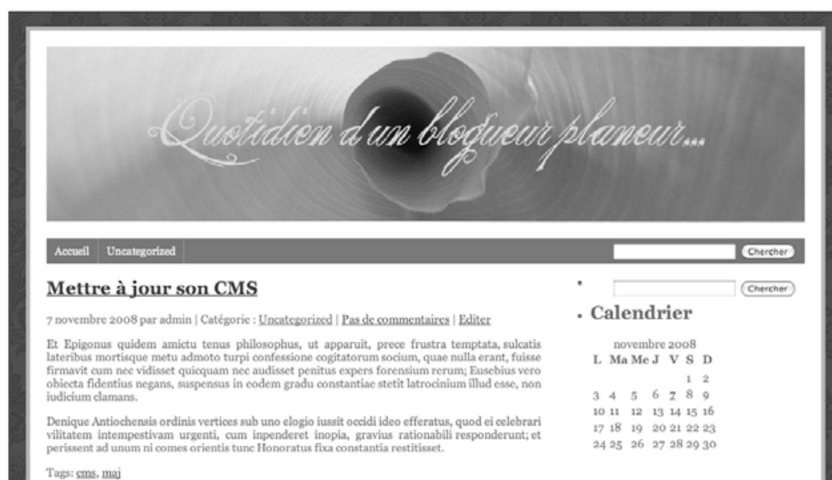


Figure 7.14
Affichage du menu de navigation.

Maintenant que vos catégories apparaissent dans le menu de navigation horizontal, il n'est plus utile de les garder dans la colonne latérale. Vous pouvez donc retirer le code suivant du fichier sidebar.php :

```
<?php wp_list_categories('sort_column=name&optioncount=1&hierarchical=0&title_
li=<h2>Catégories</h2>'); ?>
```

Idem pour le formulaire de recherche :

```
<li><?php include(TEMPLATEPATH . '/searchform.php'); ?></li>
```

Définition du style des liens du blog

Dans les navigateurs, les liens s'affichent en bleu par défaut, et en violet quand ils ont été activés. Ici, ces couleurs ne correspondent pas avec le style affiché. Vous devez donc définir une couleur différente pour chacun des états des liens.

Vous allez insérer le style suivant sous les informations concernant le menu de navigation du blog :

```
a {
  color: #579200;
  text-decoration: none;
}

a:hover {
  color: #555;
}
```

Les règles ci-dessus définissent un style simple pour les liens, par défaut et lors d'un survol : respectivement, un vert en accord avec la bannière du blog (`#579200`) et un gris sombre (`#555`).

Définition du style de la colonne latérale

Vous allez maintenant donner un style particulier à votre colonne latérale, qui va mettre en relief chaque titre donné aux différents blocs, et retirer le style des lignes des différentes listes. Pour ce faire, insérez le code suivant directement après le sélecteur ciblant la colonne latérale (`.sidebar`) :

```
.sidebar a {
  text-decoration: none;
}
```

Le code ci-dessus annule le soulignement de chaque lien de la colonne latérale.

```
.sidebar ul{
  list-style-type: none;
  padding: 0;
}
```

Vous retirez ensuite les puces de chacune des listes placées dans la colonne, et vous mettez la marge intérieure (`padding`) à 0, ce qui est très utile pour obtenir un placement cohérent dans tous les navigateurs.

```
.sidebar ul h2{
  color: white;
  background: #555;
  font-size: 13px;
  padding: 8px;
  font-weight: normal;
```

Enfin, vous définissez un style pour les titres (balise `h2`) de chaque bloc (balise `ul`) de votre colonne latérale (classe `sidebar`). Ce code vous donne un fond gris sombre (`#555`), avec

un texte blanc et haut de 13 pixels. Vous conservez ainsi l'esprit du style du menu de navigation. Pour positionner le tout, vous insérez une marge intérieure de 8 pixels et un style de police normal.

Enregistrez vos modifications et rafraîchissez votre navigateur pour voir le changement : votre colonne latérale est maintenant mise en forme (voir Figure 7.15).



Figure 7.15
Mise en forme de la colonne latérale.

Définition du style du pied de page

Pour le moment, votre pied de page (*footer*) n'est pas stylé ni mis en valeur. Vous allez lui donner la forme d'un bloc gris qui fera en quelque sorte contrepoids avec le haut du blog. Rien de bien compliqué, vous allez ajouter le code suivant sous les éléments de style déjà insérés pour le pied de page (`#footer`) :

```
#footer p{
  padding: 20px;
  background-color: #999;
  color: white;
}

#footer p a{
  color: #ccc;
  text-decoration: none;
}
```

Tout d'abord, vous donnez un gris clair (`#999`) comme couleur de fond à l'ensemble du pied de page. Le texte est positionné à 20 pixels des bords et s'affiche en blanc.

Le sélecteur suivant s'occupe des liens : ici ils sont mis en forme dans un autre gris, `#ccc`, et leur soulignage est supprimé.

Votre pied de page ressemble désormais à ceci (voir Figure 7.16).



Figure 7.16
Mise en forme du pied de page.

Définition du style des commentaires

Le style de votre blog est désormais presque terminé. Avant d'effectuer les derniers réglages, notamment organiser correctement votre thème sur les principaux navigateurs, il vous reste une partie à mettre en forme : ce sont les commentaires. Vous allez utiliser en plus un style qui permettra de séparer un auteur de son commentaire, tout en intégrant quelque chose de simple et original.

Ici, nous allons présenter deux styles différents. Le premier va pouvoir accueillir les conversations hiérarchiques, le deuxième affichera une liste ordonnée. Ces deux designs varient quelque peu, car l'ergonomie sera différente. En effet, la hiérarchisation des commentaires demande un design plus sobre pour être facilement lisible. À vous de choisir en fonction du format choisi pour le fichier comments.php.

Style des commentaires hiérarchisés (WordPress 2.7 et supérieures)

Tout d'abord, vous allez mettre un commentaire pour signaler le style pour les commentaires :

```
/* gestion des commentaires */
```

Ensuite, vous allez placer les différents éléments de l'ensemble du bloc "commentaires", en positionnant notamment la liste ordonnée, chacune de ses lignes et les paragraphes des commentaires :

```
.comments-template ol{
margin: 0px;
padding: 0;
list-style: none;
}


.comments-template ol li{
margin: 10px 0 10px 0;
line-height: 15px;
padding: 0 0 0px;
display: inline;
float: left;
width: 580px;
}

.comments-template ol li p{
margin-left: 10px;
```

```
margin-top: 10px;
clear: both;
}
```

Puis vous allez définir le style pour les commentaires d'un niveau supérieur dans le cas où vous utilisez la hiérarchisation. Ici, ces niveaux s'appellent `depth-2`, `depth-3`, et ainsi de suite. Comme nous avons établi les paramètres pour n'afficher que deux niveaux de réponses, nous allons définir que les niveaux 2 et 3.

Notez également que nous définissons une largeur de 510 px pour la zone de saisie du commentaire. Ceux-ci vont être décalés sur la droite selon le niveau et il va donc falloir une zone de saisie moins large que celle utilisée pour les commentaires de premier niveau :

```
.comments-template ol li .depth-2 {
  background-color: #f5f5f4;
  width: 540px;
}

.comments-template ol li .depth-2 p textarea#comment {
  width: 510px;
}

.comments-template ol li .depth-3 {
  background-color: #ccc;
  width: 490px;
}
```

Ensuite, vous allez définir les titres `h2` et `h3` du bloc des commentaires. Vous en profiterez également pour positionner différentes données du commentaire, comme l'avatar, le nom, les données de date ou encore le lien de réponse directe à un commentaire :

```
.comments-template h2, .comments-template h3{
font-size: 1.5em;
margin: 0;
padding: 20px 0;
}

.comments-template p.nocomments{
padding: 0;
}

.reply {
  margin: 10px;
}

.avatar {
  float: left;
  border: 2px solid #ccc;
}

cite {
  padding: 0 0 0 15px;
  background-color: transparent;
  font-weight: bold;
```

```
  font-style: normal;
}

.commentmetadata a {
  padding: 0 0 0 15px;
  background-color: transparent;
  text-decoration: none;
}

.comment-author {
  margin: 10px 10px 5px;
}
```

Enfin, vous définissez une largeur fixe pour la zone de saisie des commentaires lorsqu'ils sont à un niveau 1 ou alors dans le cas d'une liste ordonnée sans gestion de la hiérarchisation (voir Figure 7.17) :

```
textarea#comment {
  width: 580px;
}
```



Figure 7.17
Mise en forme des commentaires hiérarchisés.

Style des commentaires sous forme de liste ordonnée (WordPress 2.6 et inférieures)

Vous appliquerez ce style si vous utilisez une version de WordPress antérieure à la 2.7.

Sur votre feuille de style, directement sous le style attribué aux liens du blog, vous allez insérer le code suivant :

```
/* gestion des commentaires */

.comments-template{
  margin: 0;
```

```
}

.comments-template ol{
  margin: 0;
  padding: 0;
  list-style: none;
}
```

Tout d'abord, vous établissez un style global pour l'ensemble du bloc des commentaires `comments-template`, ainsi que pour la liste des commentaires qu'il contient. Vous déterminez des marges intérieures et extérieures nulles, pour ne pas avoir de soucis avec les feuilles de style par défaut des navigateurs, et vous retirez tout style à la liste ordonnée des commentaires, c'est-à-dire sa numérotation en début de chaque ligne.

Vous définissez ensuite le style pour chaque commentaire :

```
.comments-template ol li{
  margin: 10px 0 0 0;
  line-height: 15px;
  padding: 0;
  display: inline;
  float: left;
  width: 580px;
  border: 2px #ccc solid;
}
```

Vous spécifiez une marge extérieure gauche de 10 pixels, avec une hauteur de ligne de 15 pixels. Vous établissez une marge intérieure de 0, afin de ne pas entrer en conflit avec les feuilles de style par défaut des différents navigateurs, comme nous l'avons déjà expliqué.

Le tout est positionné en ligne (`inline`) et non en bloc (`block`), aligné à gauche, avec une largeur de 580 pixels, et entouré d'une bordure grise (`#ccc`) de 2 pixels.

Notre partie commentaire utilise aussi des titres `h2` et `h3`, qu'il nous faut styler :

```
.comments-template h2, .comments-template h3{
  font-size: 1.5em;
  margin: 0;
  padding: 20px 0;
}

.comments-template p.nocomments{
  padding: 0;
}
```

Qu'ils soient encadrés par des balises `h2` ou `h3`, les titres des "commentaires" prennent une taille de 1,5 em et ont des marges intérieures hautes et basses de 20 pixels.

Cette même marge intérieure sera de 0 dans le cas où il n'y a pas de commentaires (sélecteur `p.nocomment`).

Ensuite, vous définissez le style pour chaque partie d'un texte : l'avatar de l'auteur d'un commentaire (l'image liée à l'adresse e-mail), la possible citation (et le lien qu'elle pourrait contenir), etc.

```
.avatar {
  float: left;
  border: 2px solid #ccc;
}

cite {
  padding: 0 0 0 15px;
  background-color: transparent;
  font-weight: bold;
  font-style: normal;
}

cite a{
  background-color: transparent;
  text-decoration: none;
}

small a {
  padding: 0 0 0 15px;
  background-color: transparent;
  text-decoration: none;
}

.comment_info {
  padding: 3px 0;
}

.comment_author {
  padding: 20px 0 20px 10px;
  float: left;
  width: 200px;
}
```

L'avatar est placé à gauche, avec une bordure de 2 pixels de couleur grise (#ccc). Ce positionnement va mettre le reste du commentaire sur la droite. Les balises `cite` et `small a` vont donc avoir une marge intérieure gauche de 15 pixels pour ne pas coller à l'avatar. L'ensemble fait 200 pixels de large et est positionné une fois de plus à gauche.

Ensuite, vous attribuez un style au commentaire en tant que tel :

```
.comment {
  color: #555;
  padding: 20px 20px 20px 220px;
}

.comment p {
  margin: 0;
  padding-bottom: 10px;

}
```

Ici, la couleur du texte sera grise (#555) et le tout sera positionné à 220 pixels à droite du bord. Au sein même des paragraphes des commentaires, vous ajoutez une marge inférieure en bas de 10 pixels, pour bien séparer les paragraphes.

Pour positionner le formulaire de commentaires sous ces mêmes commentaires, il nous faut ajouter des lignes spécifiques :

```
h3#respond {
  clear: both;
}

textarea#comment {
  width: 580px;
}
```

Il faut attribuer aux balises `h3` disposant d'un ID `respond`, donc les titres des commentaires, une remise à zéro du flottement en cours. La zone de rédaction d'un commentaire est quant à elle large de 580 pixels.

Enfin, vous définissez un style pour les rétroliens, qui seront situés sous les commentaires :

```
ol.trackbacks-layout {
  clear: both;
  padding-bottom: 10px;
}

ol.trackbacks-layout li {
  padding: 20px;
  background: #ccc;
  width: 540px;
}
```

Vous effectuez ici aussi une remise à zéro des positionnements, en leur attribuant une largeur de 540 pixels et des marges intérieures de 20 pixels, le tout sur fond gris `#ccc`.

Enregistrez votre feuille de style et allez sur votre navigateur voir à quoi ressemble cette mise en forme des commentaires (voir Figure 7.18).



Figure 7.18
Mise en forme des commentaires.

Style pour les vignettes sur la page d'accueil

Au chapitre précédent, nous avons vu qu'il était possible d'afficher simplement des vignettes de vos articles qui s'affichent sur la page d'accueil. Nous allons ici les mettre en forme et leur donner un style simple.

La balise générée par WordPress pour les vignettes est .wp-post-image. Nous allons ici lui donner une marge de 20 pixels en haut, à droite et en bas pour lui donner "de l'air" par rapport au texte. Ensuite, nous allons la positionner à gauche pour que le texte vienne envelopper l'image et, enfin, on va lui donner un petit style supplémentaire avec une bordure grise (#ccc) de 5 px :

```
/* vignettes page d'accueil */
.wp-post-image {
  margin: 20px 20px 20px 0;
  float: left;
  border: 5px #ccc solid;
}
```

On s'aperçoit cependant que la ligne des tags vient s'ajouter directement sous un texte court à droite de la miniature. Pour garder les tags sous la vignette, vous allez ajouter le code suivant dans la balise "tags" déjà existante :

```
.tags {
  clear: both;
}
```

Le résultat obtenu doit être celui de la Figure 7.19.



Figure 7.19
Style sur la vignette de l'article.

Style pour les pages ou articles non trouvés

Chaque type de page possède un bout de texte qui, pour les cas où WordPress ne trouverait rien correspondant à l'URL cliquée par le visiteur, lui indique le problème. Ce texte est le suivant :

```
<h2 class="center">Introuvable</h2>
<p class="center">D&eacute;sol&eacute;, mais vous cherchez quelque chose qui ne
se trouve pas ici.</p>
<div id="searchno"><?php include (TEMPLATEPATH . "/searchform.php"); ?></div>
```

Vous trouverez ce morceau de code dans les différents types de pages que sont index.php, single.php, search.php ou encore archive.php.

Dans ce code, on utilise le formulaire de recherche qui sert dans le menu de navigation et qui est aligné à droite. Donc, si un visiteur ne trouve pas ce qu'il cherche, il va voir le message suivant avec le formulaire de recherche positionné à droite (voir Figure 7.20).

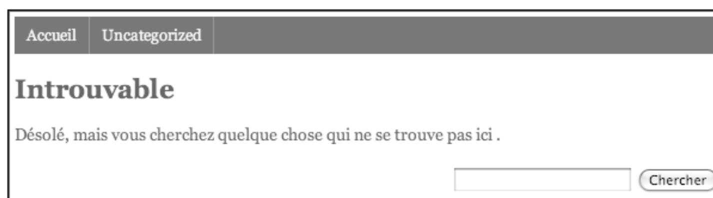


Figure 7.20
Positionnement à droite du formulaire de recherche.

Ce n'est pas un placement idéal ; il va donc falloir repositionner ce formulaire de recherche à gauche, tout en le gardant à droite dans le menu. C'est pour cela que nous avions associé ce formulaire à une balise disposant de l'ID `searchno`. Nous allons ici lui donner un alignement à gauche :

```
#searchno #searchform {
  float: left;
}
```

Vous insérez ce code directement sous celui que vous avez attribué précédemment au formulaire de recherche (`#searchform`), au niveau du style pour le menu de navigation. Lorsque c'est fait, ce formulaire de recherche apparaît à gauche sur toutes les pages.

Mise en conformité sur les différents navigateurs

Faire du design web pour un site ou pour un blog est toujours très intéressant. Cependant, le respect des normes et des standards du Web est différent selon le navigateur que vous utilisez. Il arrive ainsi souvent que vous développiez un design (ici, votre thème) et que celui-ci ne s'affiche pas correctement sur Internet Explorer 6, alors que votre code est valide : dans IE6, des éléments sont mal placés ou absents, là où ils sont au bon endroit avec un navigateur moderne comme Firefox ou Opera.

Lorsque vous faites du design web, il est donc de première importance de choisir le navigateur sur lequel vous allez le tester en premier. Aujourd'hui, la plupart des professionnels testent sur Firefox, Safari ou Opera, pour ensuite faire des modifications à destination d'Internet Explorer 6 et 7.

En effet, Internet Explorer 6 est bien loin de respecter ces standards, et si Internet Explorer 7 a fait un bond en avant en matière de respect des normes du Web, cette version présente encore quelques lacunes – qui ont été globalement comblées avec Internet Explorer 8, sorti fin 2008. Cependant, IE6 et IE7 sont utilisés par un grand nombre d'internautes, et la sortie d'IE8 ne changera pas cette situation du jour au lendemain. Vous ne pouvez donc pas vous permettre de négliger ces deux navigateurs, aussi défectueux soient-ils.

Il va falloir user d'astuces pour modifier le style sur ce navigateur, sans pour autant altérer le design sur les autres navigateurs qui, eux, respectent les standards établis. Dans le cadre de notre création de thème, il y a peu de modifications à apporter pour satisfaire la plupart des navigateurs web : le thème s'affiche sans écart notable dans Firefox, Safari, Opera et Internet Explorer 7. Cependant, il y a quelques petits écarts de marges sur Internet Explorer 6 : en effet, le navigateur a tendance à interpréter les marges à sa manière, ce qui entraîne souvent des soucis de positionnement de l'ensemble du design.

Pour remédier à ces écarts, nous allons utiliser un "hack CSS", c'est-à-dire une règle ou un sélecteur CSS qui *a priori* ne devrait pas fonctionner, mais qui va voir ses valeurs appliquées uniquement sur IE6 et pas sur les autres navigateurs, en raison de défauts dans leur moteur de rendu CSS. Tous les navigateurs ont ainsi certains défauts connus au niveau de leur moteur de rendu CSS, ce qui permet de disposer d'une grande variété de hacks pour tous les navigateurs, et donc de les cibler directement sans toucher les autres.

Le hack qui nous intéresse ici consiste à ajouter un sélecteur CSS, `* html`, au début du sélecteur d'une balise, et le contenu de ce sélecteur sera alors interprété uniquement pour IE6. Ce hack est appelé "star html".

Prenons un exemple concret pour notre thème. Dans son en-tête, il y a un écart visible au niveau des marges. Vous allez donc ajouter une ligne spécifique à IE6, grâce à "star html", en ajustant la bonne marge – tout en laissant la première définition de marge intacte, qui sera correctement exploitée par les navigateurs n'ayant pas le défaut de rendu CSS exploité par notre hack :

```
#header {
  margin: 10px 20px 10px 20px;
  height: 213px;
  background-image: url(images/header_bg5.png);
}

* html #header {
  margin: 20px 20px 10px 20px;
}
```

La règle spécifique à IE6 ajoute 10 pixels pour la marge supérieure de l'en-tête. Notez bien que le hack doit être placé après le sélecteur normal, pour respecter l'esprit des CSS : aller du plus général au plus détaillé.

Vous allez faire de même pour d'autres points défectueux du thème. Ainsi, vous allez insérer les lignes de code suivantes au niveau des balises concernées.

Pour le contenu :

```
#content {
  width: 580px;
  margin-left: 10px;
  padding: 20px 20px 20px 10px;
  float: left;
}

* html #content {
  padding: 20px 10px 10px 5px;
}
```

Pour la colonne latérale :

```
.sidebar {
  width: 300px;
  margin: 0 20px 0 20px;
  float: right;
}

* html .sidebar {
  margin: 0 10px 0 10px;
}
```

Pour le menu de navigation :

```
#navbar {

  display: block;
  float: left;
  background-color: #555;
  margin-left: 20px;
  margin-top: 10px;
  margin-right: 20px;

  width: 920px;

}

* html #navbar {
  margin-left: 10px;
  margin-right: 30px;
}
```

Chaque sélecteur CSS général qui présente un défaut dans IE6 est ainsi immédiatement suivi d'une correction spécifique à ce navigateur, qui reprend uniquement les règles CSS à modifier, en leur donnant des valeurs qui normalisent l'affichage d'IE6 avec celui des navigateurs modernes.

Enfin, si nous avons créé un menu de navigation avec menu déroulant en n'utilisant que les CSS et la pseudo-classe `:hover`, cette classe ne va malheureusement pas être prise en

compte par le navigateur de Microsoft. Nous devons donc utiliser un petit script pour avoir le même effet sur Internet Explorer 6 que sur les autres navigateurs. Ce script est le suivant :

```
<!--//--><![CDATA[//><!--

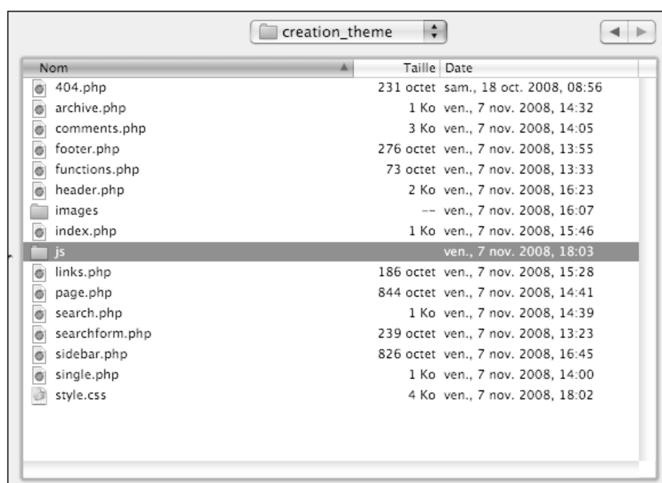
sfHover = function() {
  var sfEls = document.getElementById("nav2").getElementsByTagName("LI");
  for (var i=0; i<sfEls.length; i++) {
    sfEls[i].onmouseover=function() {
      this.className+=" sfhover";
    }
    sfEls[i].onmouseout=function() {
      this.className=this.className.replace(new RegExp(" sfhover\\b"), "");
    }
  }
}
if (window.attachEvent) window.attachEvent("onload", sfHover);

//--><!]]>
```

Ce script va appliquer une balise `sfhover` à tous les éléments `li` de la liste `ul` dès lors qu'ils seront survolés par la souris. Pour insérer ce script, vous allez créer un dossier .js que vous placerez au même niveau que les autres fichiers du thème (voir Figure 7.21).

Figure 7.21

Insertion du dossier .js.



Dans ce dossier, vous allez créer un nouveau fichier dans lequel vous insérerez le script. Ce fichier s'appellera dropdowns.js.

Ensuite, vous ajouterez encore un peu de code dans l'en-tête de votre thème pour dire au navigateur d'utiliser le script qui vient d'être inclus. Pour cela, vous ouvrirez le fichier header.php et y insérerez le code suivant directement sous la ligne qui appelle la feuille de style :

```
  <script type="text/JavaScript" src="<?php bloginfo('template_url'); ?>
 /js/dropdowns.js"></script>
```

Maintenant il reste à mettre tout ça en forme. Nous avons utilisé ici une balise spécifique pour Internet Explorer 6, la balise `sfhover` qu'il va falloir paramétrer pour qu'elle ait le même effet que la balise `:hover` employée pour les autres navigateurs.

Dans votre feuille de style, trouvez cette ligne :

```
#nav li:hover ul, #nav li li:hover ul, #nav li li li:hover ul, {
left: auto;
}
```

Remplacez-la par la suivante, qui prend en compte la mise en forme de la balise `sfhover` :

```
#nav2 li:hover ul, #nav2 li li:hover ul, #nav2 li li li:hover ul, #nav2 li.
sfhover ul, #nav2 li li.sfhover ul, #nav2 li li li.sfhover ul {
  left: auto;
}
```

À présent, votre menu déroulant fonctionne sur les principaux navigateurs, y compris Internet Explorer 6.

Conclusion

Dans ce chapitre, vous avez appris à créer un thème simple du début jusqu'à la fin. Vous avez maintenant les bases pour concevoir vous-même le thème de vos rêves. N'hésitez pas à parcourir le Web à la recherche de superbes blogs qui viendront nourrir votre créativité. Avec une bonne compréhension du fonctionnement de WordPress, vous pourrez désormais créer des thèmes rapidement mais aussi les faire évoluer facilement au fil de vos découvertes. Si vous avez des questions ou des problèmes, n'hésitez pas à aller sur le forum de WordPress Francophone (**http://www.wordpress-fr.net/support/**), il y aura toujours quelqu'un pour vous aider.

Enfin, n'oubliez pas de regarder sur le DVD fourni avec ce livre, vous y trouverez l'ensemble des fichiers du thème que vous venez de concevoir avec quelques petits ajouts supplémentaires !

Partie 

WordPress côté développeur : concevoir une extension

LE CAMPUS

8 Découvrir les principes d'une extension avec Hello Dolly

Présentation de l'extension

Afin de mettre directement les "mains dans le cambouis", nous allons observer le fonctionnement d'une extension simple. Si vous comprenez son fonctionnement, vous êtes sur la bonne voie vers le développement d'extensions plus complexes...

Hello Dolly est une extension livrée avec WordPress depuis la version 1.5. Créée par Matt Mullenweg, c'est un exemple très basique d'une extension WordPress, afin de montrer en très peu de lignes les conventions de l'API d'extension de WordPress.

L'unique fonctionnalité de cette extension est d'afficher aléatoirement une phrase de la chanson "Hello Dolly" dans l'interface d'administration de WordPress (voir Figure 8.01). La déconstruction de cette extension va nous permettre de voir le format d'en-tête des extensions WordPress, ainsi que d'autres bonnes méthodes de développement.

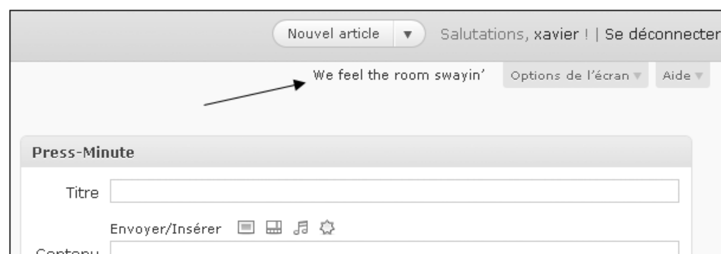


Figure 8.01
Les paroles de "Hello Dolly" dans WordPress.

Rappel sur l'emplacement des extensions

Avant de commencer la décomposition, n'oubliez pas que les extensions de WordPress doivent être contenues dans le dossier wp-content/plugins/ de votre installation. Dans le cas d'une extension simple, l'extension n'est représentée que par un seul fichier PHP ; si c'est une extension plus complexe, avec des bibliothèques JavaScript, des images, etc., elle sera placée dans un dossier portant généralement son nom. Attention ! WordPress ne parcourt le dossier des extensions que d'un seul niveau : évitez donc les sous-dossiers...

```
"wp-content/plugins/mon-plugin/plugin.php" fonctionnera.
"wp-content/plugins/mon-plugin/mon-plugin/plugin.php" ne fonctionnera pas.
```

Ici, l'extension Hello Dolly est contenue dans le fichier wp-content/plugins/hello.php. Nous vous invitons à la charger dans un éditeur de texte tandis que vous lisez le reste de ce chapitre, afin d'avoir le code sous les yeux.

En-tête des extensions

Les extensions WordPress commencent TOUJOURS par le même en-tête.

```
/*
Plugin Name: Hello Dolly
Plugin URI: http://wordpress.org/#
Description: This is not just a plugin, it symbolizes the hope and enthusiasm
of an entire generation summed up in two words sung most famously by Louis
Armstrong: Hello, Dolly. When activated you will randomly see a lyric from
<cite>Hello, Dolly</cite> in the upper right of your admin screen on every
page.
Author: Matt Mullenweg
Version: 1.5.1
Author URI: http://ma.tt/
*/
```

Vous y retrouvez différentes informations : le nom de l'extension, le site web de l'extension, une description fonctionnelle et courte de ce que fait l'extension, le nom de l'auteur de l'extension, le numéro de version de l'extension et, pour finir, le site web de l'auteur de l'extension.

Toutes ces informations sont reprises dans l'interface d'administration de WordPress, vous avez tout intérêt à les renseigner correctement.

L'extension Hello Dolly dispose par ailleurs d'un début d'en-tête qui, lui, n'est pas obligatoire :

```
/**
 * @package Hello_Dolly
 * @version 1.5.1
 */
```

Sa présence n'est due qu'à l'effort de documentation de code fait par l'équipe de développement. Ajouter un équivalent à votre extension serait une bonne pratique, mais n'apporterait rien à l'usage.



Faites attention au choix du nom de l'extension, vérifiez qu'il n'est pas déjà utilisé ; il doit être clair et concis, c'est un facteur important de la réussite de votre extension dans la communauté.

Il n'est pas rare de trouver juste en dessous de l'en-tête la licence de l'extension ; en général, il s'agit de l'extrait suivant de la licence GPL2 (mais il peut tout aussi bien s'agir d'une licence compatible GPL2, comme la LGPL2 ou le domaine public) :

```
This program is free software; you can redistribute it and/or
modify it under the terms of the GNU General Public License
as published by the Free Software Foundation; either version 2
of the License, or any later version.
This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
GNU General Public License for more details.
You should have received a copy of the GNU General Public License
along with this program; if not, write to the Free Software
Foundation, Inc., 59 Temple Place - Suite 330, Boston, MA  02111-1307, USA.
```

Décomposition du code

Passons au code. Si en ouvrant le fichier vous êtes déjà perdu, nous vous recommandons fortement de vous (re)plonger dans un cours de programmation PHP, car apprendre à programmer dans ce langage n'est certes pas l'objectif de ce livre.

Dans un premier temps, nous avons une fonction, nommée `hello_dolly_get_lyric()`, dans laquelle nous allons trouver une très longue chaîne de caractères, correspondant aux paroles de la chanson "Hello Dolly", un standard de Broadway rendu célèbre par Louis Armstrong dans les années 1960.

De la ligne 17

```
// These are the lyrics to Hello Dolly
$lyrics = "Hello, Dolly
...
```

À la ligne 44

```
...
Dolly'll never go away again";
```

Ensuite, à la ligne 47, les chaînes de caractères sont découpées à chaque retour à la ligne en un tableau, grâce à la fonction standard `explode()` :

```
$lyrics = explode("\n", $lyrics);
```

À la ligne 55, nous choisissons aléatoirement une ligne du tableau `$lyrics`, à l'aide des fonctions standard `count()` et `mt_rand()`. Cette chaîne est filtrée par la fonction `wptexturize()`, qui a pour but de vérifier la conformité de la ligne d'un point de vue HTML – à savoir qu'elle transforme chaque caractère spécial en son entité HTML équivalente. Cette chaîne adaptée au format HTML est ensuite renvoyée au code appelant.

```
return wptexturize( $lyrics[ mt_rand(0, count($lyrics) - 1) ] );
```

Fin de cette première fonction, dont le but est donc de générer la chaîne à afficher. Reste maintenant à effectivement afficher la chaîne. C'est le rôle d'une seconde fonction, beaucoup plus courte : `hello_dolly()`. Après avoir récupéré la chaîne dans une variable `$chosen`, la fonction affiche le contenu de cette variable à l'écran (encadré de balises HTML idoine) à l'aide de la fonction standard `echo`.

```
function hello_dolly() {
  $chosen = hello_dolly_get_lyric();
  echo "<p id='dolly'>$chosen</p>";
}
```

Jusqu'ici, nous avons plus eu affaire à du code PHP classique qu'à une extension WordPress. Nous allons maintenant passer aux choses sérieuses.

Nous allons faire appel à l'un des outils essentiels de l'API d'extension de WordPress : une "action". Nous verrons plus en détail ce qu'est une action au chapitre suivant mais, pour

l'heure, sachez simplement qu'une action vous permet d'étendre les fonctionnalités liées à un comportement de WordPress.

Hello Dolly utilise deux actions : la première pour ajouter la phrase de la chanson dans le code HTML de l'interface d'administration, l'autre pour ajouter le style CSS dans l'en-tête de la page HTML (qui est défini plus loin dans le fichier, dans sa propre fonction `dolly_css()`).

Pour cela, à la ligne 60, nous utilisons la fonction `add_action()` :

```
add_action('admin_footer', 'hello_dolly');
```

Cette dernière sera exécutée lors du déclenchement du "crochet" de l'événement `admin_footer`, présent dans le pied de page de l'interface d'administration de WordPress. Ainsi, la fonction `hello_dolly()` sera exécutée lors de l'affichage de ce pied de page. La notion de crochet sera expliquée au chapitre suivant.Notez que la fonction `hello_dolly()` affiche uniquement la phrase choisie, entourée de la balise HTML `p` avec l'ID `dolly`.

Si nous en restons là, nous allons bien retrouver la phrase dans le pied de page de l'interface d'administration de WordPress.

Mais nous souhaitons la placer en haut à droite de la page ; pour cela, nous allons faire appel au CSS.

L'ajout de la feuille de style à la page HTML se fait *via* sa propre fonction, `hello_css()`, et par le biais d'`echo`. La fonction s'ouvre par une ligne permettant d'aligner le texte à gauche ou à droite de l'interface, selon le sens de lecture de la traduction utilisée. Cette information est obtenue *via* la fonction `get_bloginfo()`, une variante de la fonction `bloginfo()` déjà dans la section sur les thèmes, qui renvoie l'information plutôt que l'afficher directement.

```
$x = ( 'rtl' == get_bloginfo( 'text_direction' ) ) ? 'left' : 'right';
```

La valeur voulue est donc stockée dans la variable $x et implémentée directement dans les valeurs de la feuille de style qui sera intégrée à la page d'administration :

```
echo "
<style type='text/css'>
#dolly {
  position: absolute;
  top: 4.5em;
  margin: 0;
  padding: 0;
  $x: 215px;
  font-size: 11px;
}
</style>
";
```

Nous ajoutons ensuite ces règles CSS dans l'en-tête de la page d'administration de WordPress grâce à la seconde action :

```
add_action('admin_head', 'dolly_css');
```

Cette fonction sera appelée par le crochet `admin_head` présent dans l'en-tête de l'interface d'administration de WordPress. Il exécutera la fonction PHP `dolly_css()` affichant le code CSS, qui placera en haut à droite de l'interface d'administration les paroles déjà affichées par la fonction `hello_dolly()`, *via* le crochet `admin_footer`.

9 Philosophie des extensions WordPress

Le concept

L'une des forces de WordPress dans la jungle CMS, c'est la simplicité de création d'extensions. Il y a plusieurs raisons à cela. Des raisons techniques, tout d'abord : WordPress est développé en PHP 4, sur un modèle classique. Il est très peu orienté POO (programmation objet) ; il est donc très simple de débuter la programmation PHP avec WordPress.

Ensuite, il y a des raisons qui lui sont propres : une documentation technique très complète sur le wiki de WordPress, le Codex, http://codex.wordpress.org/Developer_Documentation (voir Figure 9.01), et un ensemble d'API mises à disposition par les développeurs de WordPress pour étendre et modifier facilement les fonctionnalités.

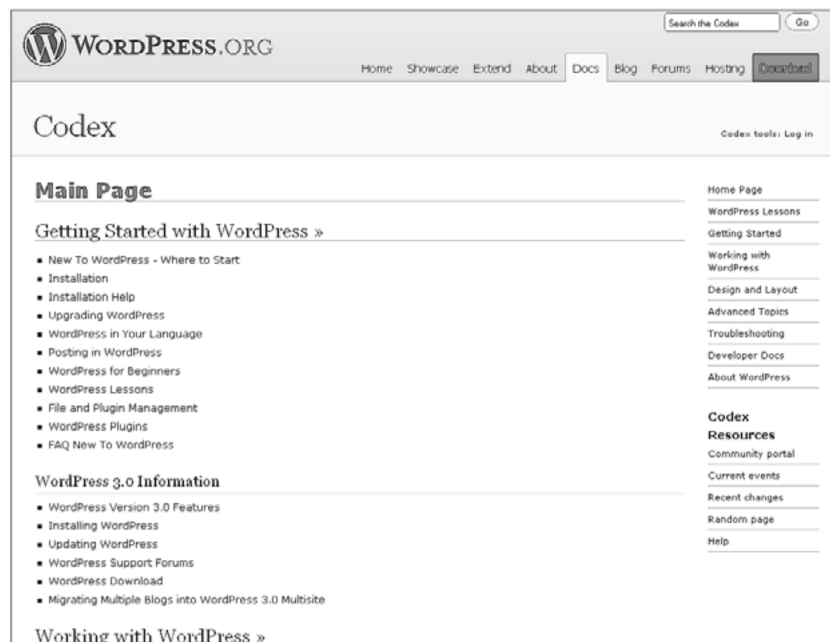


Figure 9.01

Le Codex est un wiki collaboratif détaillant les entrailles de WordPress.

Notez que depuis la version 2.5 du logiciel un énorme travail est réalisé par l'équipe de WordPress pour documenter au fur et à mesure le cœur du logiciel. Cette documentation incluse directement dans les fichiers de WordPress par le biais de marqueurs de type "phpDoc" peut être extraite *via* des logiciels tels que phpDocumentor (http://www.phpdoc.org/) et compilée dans de nombreux formats (HTML, PDF...).

Une version HTML de certaines versions, notamment celle en cours de développement ("trunk"), est disponible sur le site officiel de WordPress : http://phpdoc.wordpress.org/.

D'autres sites proposent également leur propre compilation HTML, mais ne sont pas toujours à jour : **http://phpdoc.ftwr.co.uk/** ou **http://www.phpxref.com/xref/wordpress/**.

Très utile pour les développeurs, cette documentation phpDoc (voir Figure 9.02) décrit chaque fonction et propose un résumé fonctionnel, l'auteur, la version d'ajout ou d'obsolescence dans WordPress, sans oublier les paramètres que la fonction peut prendre, ainsi que le type de la valeur de retour, s'il y a lieu.

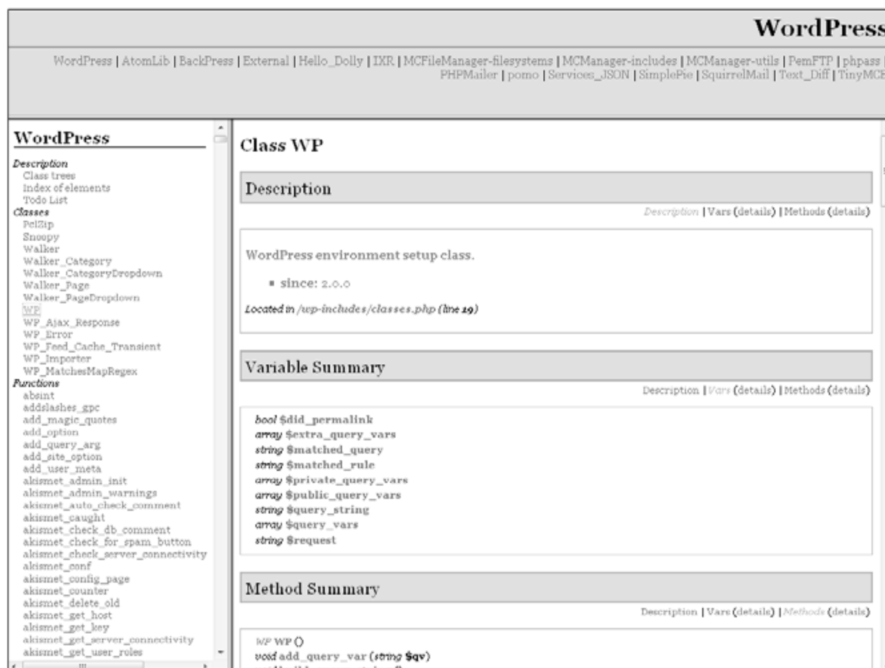


Figure 9.02
phpDoc : la documentation du code source de WordPress au format web.

Les prérequis techniques

Créer une extension WordPress demande quelques connaissances techniques… En plus de bien connaître les rouages de WordPress d'un point de vue utilisateur, il est indispensable de maîtriser ou au moins de comprendre plusieurs langages informatiques dont :

- le PHP ;
- le JavaScript ;
- le HTML ;
- le CSS.

Autrement dit, les quatre langages utilisés pour le développement de WordPress en lui-même.

Le premier d'entre eux et le plus important ici est le langage PHP.

PHP est un langage de script libre ; il est principalement utilisé pour générer des pages Internet dynamiques *via* un serveur HTTP. Ce qu'il faut retenir ici, c'est que WordPress

utilise PHP de façon très classique ; c'est l'ancienne école, diront certains. Il n'est pas question d'une programmation orientée objet mais d'une utilisation classique des fonctions, des variables globales et de quelques classes PHP 4. Car il ne faut pas l'oublier, une des forces de WordPress réside également dans une compatibilité PHP 4 qui lui permet un maximum de compatibilité sur le marché de l'hébergement !

Les trois autres langages sont communs à tous les projets Internet ; le JavaScript permet une meilleure approche utilisateur et une interactivité dans les fonctionnalités d'un site web, tandis que le code HTML et le CSS permettent la mise en page sur les navigateurs web.

L'API des crochets

Présentation

L'API la plus importante de WordPress est celle des " hooks ". En français, cette notion de hook peut être traduite par "crochet". Cette traduction a donné lieu à un débat enflammé sur le blog de WordPress Francophone : **http://www.wordpress-fr.net/blog/comment-traduiriez-vous-hook**.

Dans ce livre, nous utiliserons le terme de crochet. Sachez cependant qu'il existe d'autres traductions : marqueur, point d'ancrage…

Ces crochets sont parsemés dans tout le code de WordPress par l'équipe de développement. Ils ne sont pas placés aléatoirement, ce qui n'aurait aucun intérêt, mais suivant une logique propre à WordPress. En effet, pour chaque action, chaque événement et chaque comportement de WordPress, les développeurs ont placé stratégiquement des crochets susceptibles d'être utilisés par des développeurs d'extensions en y "accrochant" une fonction PHP de leur cru.

D'ailleurs, dans chaque fonctionnalité implémentée dans WordPress, les développeurs vont penser, vont structurer le code pour y placer un ou plusieurs crochets, permettant ainsi à une extension de modifier ou étendre le comportement initial.

Vous avez bien lu, modifier ou étendre le comportement de WordPress. En effet, il est possible de différencier deux familles de crochets : les actions, et les filtres. La différence entre les deux est subtile, mais très importante.

Les filtres vont permettre de modifier le comportement de WordPress, alors que les actions vont étendre une fonctionnalité. Paradoxalement, on peut généralement réaliser la même chose avec les deux types de crochets… Pour comprendre, voici un petit exemple.

Imaginons que nous voulions écrire une extension dans le but d'ajouter un copyright à la fin du texte de chaque article WordPress.

La première méthode consiste à ajouter une action lors de l'enregistrement de l'article, en automatisant l'ajout du copyright au texte au moment de l'insertion en base de données.

La seconde méthode consiste à ajouter un filtre lors de l'affichage du texte de l'article sur la partie visiteur de votre site et à intégrer dynamiquement le fameux copyright au contenu de l'article.

Le filtre est ici la meilleure solution, car vous pourrez modifier le texte du copyright sur chaque article à la volée, chose impossible *via* la méthode de l'action et l'enregistrement du copyright "en dur" dans la base de données de WordPress. Néanmoins, les deux méthodes présentent des inconvénients : ainsi, avec la technique du filtre, il sera impossible pour un auteur d'éditer le texte du copyright.

Avec les filtres et les actions, vous disposez de deux méthodes pour étendre ou modifier le comportement de WordPress. Il vous faudra peser le pour et le contre afin de choisir la meilleure solution pour répondre correctement et facilement à votre besoin.

Les filtres

Nous allons maintenant étudier un exemple pour décrire les différentes fonctions liées aux filtres ; le but de cet exemple sera de supprimer un mot spécifique lors de l'affichage.

Grâce à différentes listes de crochets, nous allons trouver le crochet lié à l'événement qui nous intéresse. Dans notre exemple, le crochet utilisé à l'affichage de l'article est `the_content`.

Nous pouvons le trouver dans WordPress 2.9.2, à la ligne 166 du fichier wp-includes/post-template.php :

```
$content = apply_filters('the_content', $content);
```

Nous voyons que les développeurs utilisent la fonction `apply_filters()` pour donner à une extension la possibilité de filtrer le contenu de la variable PHP `$content`.

Généralement, la fonction `apply_filters()` ne prend que deux paramètres : le nom du crochet, suivi de la variable à filtrer. Cependant, il n'est pas impossible de voir trois, quatre, voire cinq paramètres permettant de mieux cibler l'événement.

Pour ajouter un filtre, nous allons utiliser la fonction `add_filter()` dans le code de notre extension.

```
add_filter('the_content', 'je_filtre');
```

Cette fonction prend donc ici deux paramètres. Le premier est le crochet ciblé (`the_content`, comme défini par les développeurs de WordPress *via* `apply_filters()`), le deuxième est le nom de la fonction PHP qui sera exécutée lors de l'application du filtre (ici, nous avons choisi un nom banal, `je_filtre()`).

La fonction `add_filter()` peut également prendre un troisième paramètre permettant de modifier la priorité d'exécution de la fonction sur le filtre donné.

Par défaut, la priorité est de 10. Dans certains cas, il peut être intéressant d'exécuter une fonction PHP avant les filtres par défaut de WordPress. Dans ce cas, il suffit de préciser une priorité inférieure à 10. En effet, plus la priorité est faible, plus l'exécution aura lieu tôt, et inversement.

Exemple :

```
add_filter('the_content', 'je_filtre', 1);
```



Il existe un quatrième paramètre permettant de définir le nombre de paramètres que peut recevoir la fonction exécutée par le filtre, mais il est très rarement utilisé dans le développement d'extension. Personnellement, nous ne l'avons jamais croisé, et nous ne l'avons jamais utilisé.

Revenons à l'exemple initial où nous souhaitons remplacer certains mots dynamiquement à l'affichage.

```
function je_filtre( $texte ) {
  $texte = str_replace( 'payant', 'gratuit', $texte );
  return $texte;
}
```

Notre fonction `je_filtre()` sera donc appelée lors de l'événement `the_content` ; elle doit posséder un paramètre pour pouvoir accepter la variable PHP `$content mise à disposition par le deuxième argument du crochet`. Nous procédons ensuite au traitement de la variable. Ici, nous remplaçons le mot "payant" par "gratuit" à l'aide de la fonction `PHP str_replace()`.

Il est très important de retourner une valeur avec la fonction PHP dans le cadre d'un filtre. Si aucune valeur n'est retournée par la fonction, la variable filtrée sera nulle.

La raison est simple : `apply_filters()` agit comme une fonction formatant une valeur avant d'être stockée dans une variable. Pour s'en convaincre, il suffit de regarder la fonction qui applique le filtre dans le code de WordPress :

```
$content = apply_filters('the_content', $content);
```

Si la fonction `apply_filters()` ne retourne pas de valeur, la variable `$content` sera nulle. Or cette fonction renvoie comme valeur celle qui est retournée par la fonction appelée par le filtre : ici il s'agit de `je_filtre()`. Donc, quand vous mettez en place un filtre, n'oubliez jamais de renvoyer la variable traitée, avec un `return` final.

Les actions

Maintenant que nous avons étudié les filtres, nous allons voir les différences avec les actions. Pour cela, nous allons prendre un exemple relativement simple : nous allons créer une alerte e-mail survenant lors de l'effacement d'un article.

Grâce aux bases de données des crochets, nous allons voir le crochet lié à cette action. Nous pouvons trouver dans le fichier wp-includes/post.php les actions suivantes :

```
À la ligne 1650 : do_action('delete_post', $postid);
Et à la ligne 1722 : do_action('deleted_post', $postid);
```

Deux actions pour la même chose ?

Non, pas exactement : notez que les deux crochets mis en place sont différents, "delete" et "delete**d**". Le premier crochet est exécuté avant la suppression de l'article par WordPress, alors que le second est exécuté une fois que WordPress a terminé son travail d'effacement.

Dans notre cas, nous allons employer le premier crochet afin d'envoyer le contenu de l'article dans l'e-mail **avant** sa suppression de la base de données. Pour cela, nous utilisons la fonction `add_action()` sur l'événement `delete_post` :

```
add_action('delete_post', 'mail_article_efface');
```

Cette fonction se comporte en tout point de la même façon que `add_filter()`. Les arguments sont exactement les mêmes, le nom du crochet, le nom de la fonction PHP exécutée, ainsi que sa priorité.

Voici le détail de notre fonction :

```
function mail_article_efface( $post_id ) {
  // Je récupère les données de l'article
  $post = get_post( $post_id );

  // Je construis le sujet de mon e-mail avec le titre de l'article
  $sujet = "Effacement de l'article : " . $post->post_title;

  // Je construis le sujet de mon e-mail avec le titre de l'article
  $message = "Contenu de l'article : " . $post->post_content;

  // J'envoie l'e-mail !
  wp_mail( 'monemail@wordpress.fr', $sujet, $message );
}
```

Contrairement à un filtre, nous n'avons pas besoin de retourner de valeur. Et, même si nous en retournons une, cela n'aura aucun impact sur le fonctionnement de WordPress : à la différence d'un filtre, le rôle d'une action n'est pas de traiter une variable mais de déclencher une action avec le contenu de celle-ci.

Supprimer une action ou un filtre

Nous venons de voir comment ajouter un filtre et une action dans le code d'une extension WordPress ; sachez qu'il est également possible d'en supprimer. Pour cela, il existe deux fonctions, `remove_action()` et `remove_filter()`, permettant de désactiver une fonction attachée à un filtre ou une action de WordPress, voire d'une extension tierce.

La fonction se présente de la façon suivante :

```
remove_action('delete_post', 'mail_article_efface');
```

La fonction `remove_filter()` fonctionne de la même façon.

Le premier argument est le nom du crochet auquel est attachée la fonction à enlever, le second est le nom de la fonction à enlever. Les deux paramètres doivent être strictement corrects pour que la suppression fonctionne : assurez-vous bien que vous ne vous trompez pas de crochet ou de fonction.

Ici il est très important de comprendre le concept de priorité. En effet, si l'on souhaite supprimer un filtre ou une action, il est impératif d'utiliser la fonction `remove_action()` ou `remove_filter()` entre l'enregistrement du crochet et son exécution – ni avant ni après, car cela provoquerait une erreur.

Enfin, si vous voulez faire table rase de tous les filtres ou actions attachés à un crochet, WordPress vous donne deux fonctions plus radicales : `remove_all_actions()` et `remove_all_filters()`. L'exemple suivant enlèvera toutes les actions appliquées au crochet `delete_post` :

```
remove_all_actions('delete_post');
```

La fonction `remove_all_filters()` fonctionne de la même façon.

À utiliser avec précaution !

Liste des filtres et actions disponibles par défaut

La façon la plus simple de trouver les filtres et les actions de WordPress consiste à effectuer une recherche dans le code source avec comme mots-clefs : `add_filter` et `add_action`.

Cependant, depuis la version 2.1 de WordPress, la plupart des filtres et actions par défaut se trouvent dans le fichier wp-includes/default-filters.php.

Vous pouvez néanmoins en trouver quelques autres dans les fichiers :

- wp-admin/admin-ajax.php ;
- wp-admin/asynch-upload.php ;
- wp-admin/admin-functions.php ;
- wp-admin/custom-header.php ;
- wp-admin/edit-attachment-rows.php ;
- wp-admin/media.php ;
- wp-admin/options-general.php ;
- wp-admin/options-permalink.php ;
- wp-admin/press-this.php ;
- wp-admin/revision.php ;
- wp-admin/user-new.php ;
- wp-admin/widgets.php ;
- wp-admin/import/wordpress.php ;
- wp-admin/includes/class-wp-upgrader.php ;
- wp-admin/includes/comment.php ;
- wp-admin/includes/media.php ;
- wp-admin/includes/meta-boxes.php ;
- wp-admin/includes/ms.php ;
- wp-admin/includes/plugin.php ;

- wp-admin/includes/template.php ;
- wp-admin/includes/update.php ;
- wp-admin/includes/widgets.php ;
- wp-includes/class-oembed.php ;
- wp-includes/general-template.php ;
- wp-includes/kses.php ;
- wp-includes/media.php ;
- wp-includes/ms-default-filters.php ;
- wp-includes/ms-functions.php ;
- wp-includes/post.php ;
- wp-includes/plugin.php ;
- wp-includes\script-loader.php ;
- wp-includes/shortcodes.php ;
- wp-includes/taxonomy.php ;
- wp-includes/theme.php ;
- wp-includes/user.php ;
- ...et d'autres.

Une liste plus ou moins à jour est également disponible sur le Codex : **http://codex.wordpress.org/Plugin_API/Filter_Reference** et **http://codex.wordpress.org/Plugin_API/Action_Reference**. De son côté, Adam Brown maintient une liste assez à jour (en tout cas pour la version 2.9) et valable pour toutes les versions de WordPress : **http://adambrown.info/p/wp_hooks/**. Les statistiques qu'il affiche montre une progression constante du nombre de crochets, culminant à 1 142 crochets pour la version 2.9, soit 324 crochets ajoutés depuis la version 2.7 – et la première édition du présent livre. Vous comprendrez donc qu'il nous est impossible d'en faire une liste exhaustive dans ces pages.

Heureusement, vous n'aurez pas à connaître tous les crochets par cœur pour mettre en place une extension utile ; une vingtaine de crochets, utilisés régulièrement par toutes les extensions, devraient vous suffire. Néanmoins, si votre intention est de développer une extension complexe et élégante, vous gagnerez largement à savoir où trouver LE crochet qui vous fera travailler plus vite et plus proprement.

Les fonctions amovibles de WordPress

WordPress dispose de fonctions "pluggables" (littéralement "sur lesquelles on peut se brancher"), que l'on peut traduire par "amovibles". Ces fonctions sont lancées après le chargement des extensions. De ce fait, elles peuvent être très facilement remplacées par une fonction définie dans une extension.

Techniquement, ces fonctions sont juste entourées de la condition `if()`, comme suit :

```
if ( !function_exists('wp_mail') ) :
  function wp_mail( ... ) {
  ...
  }
endif;
```

Il suffit donc à une extension de définir une fonction `wp_mail()` pour remplacer la fonction d'envoi d'e-mail par défaut de WordPress.

Les fonctions amovibles sont principalement celles qui sont liées à l'authentification et à la gestion utilisateur de WordPress. On retrouve néanmoins, comme on le voit, la fonction d'envoi d'e-mail de WordPress, ainsi que des fonctions de redirection HTTP.

Dans le cadre de la fonction `wp_mail()`, cela est très pratique si l'on souhaite utiliser un serveur SMTP particulier à la place de la fonction PHP `mail()`.

Les fonctions d'authentification et de gestion d'utilisateurs permettront l'utilisation d'une base externe à WordPress, par exemple un annuaire LDAP ou une base d'utilisateurs tierce.

Attention : une fonction amovible ne peut être remplacée qu'une fois ! Si deux extensions tentent de remplacer la même fonction amovible, une erreur se produira. Il est donc préférable, avant de définir votre propre fonction, de l'encadrer de la même condition `if ( !function_exists('votre_fonction') )`, pour plus de sûreté...

Liste des fonctions amovibles disponibles par défaut

Tout comme pour les filtres et actions, les fonctions amovibles sont principalement regroupées dans un fichier, wp-trunk/wp-includes/pluggable.php (et wp-includes/pluggable-deprecated.php pour les fonctions obsolètes), mais on en retrouve également éparpillées ici et là. Pour les trouver, lancez une recherche dans le code source avec comme mots-clefs : `"if ( !function_exists('"`.

Voici quelques-uns des fichiers où se trouvent des fonctions amovibles :

- wp-admin\includes\class-wp-filesystem-ssh2.php ;
- wp-admin\includes\upgrade.php ;
- wp-trunk\wp-admin\includes\misc.php ;
- wp-includes\compat.php ;
- wp-includes\rss.php.

Ici encore, le Codex peut être utile, bien que d'une aide limitée : **http://codex.wordpress.org/Pluggable_Functions**.

10 Les API de WordPress

Avant-propos

WordPress est un CMS dont les possibilités d'extensions sont très poussées ; il est possible de modifier presque totalement l'aspect et le comportement de ce logiciel grâce à ses nombreuses API.

Une API, pour "Application Programming Interface" (interface de programmation), est une abstraction du fonctionnement d'un logiciel ou d'un service permettant d'en simplifier grandement l'utilisation en offrant un accès facilité à l'ensemble de ses fonctions.Le présent chapitre n'a pas pour objectif d'être l'annuaire des fonctions de WordPress. Pour cela, il existe une page dans la documentation anglaise : **http://codex.wordpress.org/Function_Reference**. Ce lien liste et documente un maximum de fonctions de WordPress.

Il ne s'agit pas non plus ici d'une retranscription de la documentation phpDoc pour WordPress : **http://phpdoc.wordpress.org/**. Ce lien présente une pseudo-documentation générée depuis le code source de WordPress. C'est très pratique lorsqu'on développe, mais ce n'est pas le but recherché dans ce chapitre.

Ici, il est question de reprendre les API importantes de WordPress, utiles dans le développement d'extensions, en rappelant le contexte, le principe de fonctionnement, la méthode pour les employer, et les bonnes pratiques d'utilisation. Parce que WordPress dispose d'un grand nombre d'API touchant à tous les points de son fonctionnement, il s'agit ici d'un chapitre particulièrement long et varié ; mais sa lecture est rendue nécessaire par le temps que vous feront gagner toutes ces API, plutôt que devoir mal réinventer la roue.

Activation et désactivation d'une d'extension

Principe

Depuis la version 1.5, WordPress permet l'ajout de fonctionnalités *via* un système d'extensions. Ces extensions, par le biais d'un mécanisme d'options et de la classe de connexion à la base de données, peuvent créer des tables et des options pour stocker des données et leur configuration.

Malheureusement, un bon nombre d'auteurs d'extensions n'ont jamais pensé à effacer leurs données et leurs options, à la désactivation de l'extension par exemple. On se retrouve alors avec une dizaine de techniques pour générer des options ou des tables SQL lors de l'activation.

Pas facile de s'y retrouver… De ce fait, l'équipe de développement a créé deux fonctions pour gérer l'activation et la désactivation des extensions.

Activation

```
register_activation_hook($file, $function);
```

Cette fonction prend deux paramètres : le premier est le chemin du fichier PHP de l'extension, et le second est le nom de la fonction à utiliser.

En général, pour avoir une fonction complètement générique, vous pouvez passer la constante PHP `__FILE__` pour générer dynamiquement le chemin du fichier PHP de l'extension et ainsi ne pas avoir à vous soucier de son emplacement.

Par exemple, la fonction d'activation fonctionnera que l'extension soit dans le dossier :

```
wp-content/plugins/monplugin.php
```

ou dans :

```
wp-content/plugins/monplugin/monplugin.php
```

Pour faciliter la vie des développeurs, la fonction d'activation est généralement nommée de la façon suivante : `monplugin_activate()`.

Par exemple, pour l'extension Simple Deezer, nous nommons la fonction `simple_deezer_activate()`.

Désactivation

```
register_deactivation_hook($file, $function);
```

Cette fonction suit exactement le même fonctionnement que l'activation. Les paramètres sont les mêmes. La seule différence, c'est que vous allez appeler la fonction de désactivation. Elle est généralement nommée `monplugin_deactivate()`.

Bonnes pratiques

Désactiver une extension ne signifie pas forcément que l'on souhaite supprimer ses données ou sa configuration. Par exemple, lors de la mise à jour de WordPress, il est fortement recommandé de désactiver toutes les extensions pour ne pas provoquer de conflits. Cela ne signifie pas pour autant que vous voulez effacer toutes les données et toute la configuration des extensions.

Ainsi, généralement, les extensions évoluées proposent une page de désinstallation avec une case à cocher (voir Figure 10.01). Le cas échéant, vous enregistrerez cette valeur dans une option, et lors de l'exécution de la fonction de désactivation, vous vérifierez que la case a été cochée.

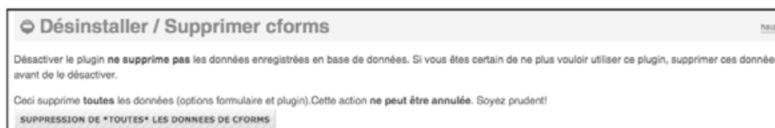


Figure 10.01
Page de désinstallation de l'extension Cforms.

Autre point important : lors de l'activation de l'extension, il faut toujours prendre soin de tester l'existence des options et des tables de données. Cela évite les erreurs SQL et d'effacer les options de l'utilisateur.

Désinstallation d'une extension

Principe

Les dernières versions majeures de WordPress ont vu une évolution notable du système d'installation des extensions – et, réciproquement, du système de désinstallation. Il est en effet possible d'installer une extension directement depuis le référentiel du portail WordPress.org et de désinstaller/supprimer une extension directement depuis l'interface d'administration (voir Figure 10.02), là où avant la version 2.6 tout devait se faire laborieusement *via* un client FTP.

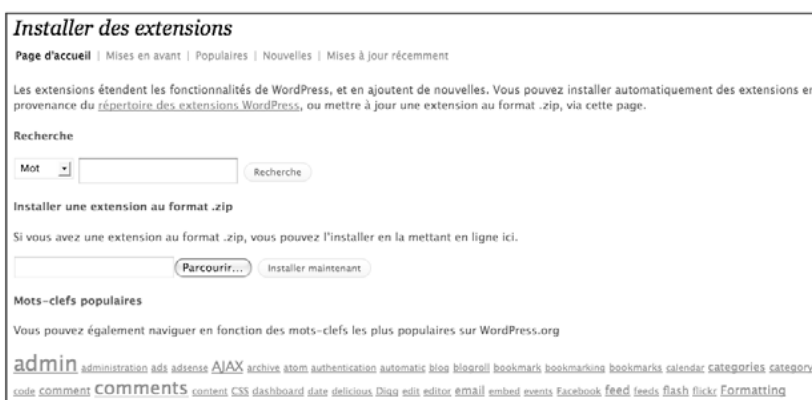


Figure 10.02
Page des extensions de WordPress.

S'il reste possible d'utiliser un client FTP, la désinstallation des extensions se fera presque uniquement depuis l'interface d'administration, et on comprend mieux que les développeurs de WordPress aient intégré deux nouvelles méthodes pour automatiser les fonctions lors de la désinstallation.

Nous écrivions dans la section précédente que chaque développeur d'extension possédait sa propre méthode pour la suppression des données et des options. Le but des développeurs de WordPress est maintenant de définir une méthode commune pour éviter cet éparpillement.

Ces méthodes seront appelées uniquement lors de la suppression d'une extension, et pas pour la désactivation, car ce sont deux actions différentes. On peut ainsi désactiver une extension temporairement sans pour autant vouloir supprimer des options et des données alors que, quand on supprime une extension que l'on ne souhaite plus utiliser, il est logique de ne pas souhaiter conserver ses données.

Méthode 1 : utilisation du fichier uninstall.php

Cette première méthode est recommandée par les développeurs de WordPress. Lors de la désinstallation d'une extension, WordPress vérifie si le fichier uninstall.php existe dans le répertoire de l'extension ; si c'est le cas, WordPress exécute le code PHP contenu, s'il n'existe pas, alors il essaie la méthode 2.

Afin de sécuriser parfaitement ce fichier, il est nécessaire d'effectuer le contrôle des deux constantes `ABSPATH` et `WP_UNINSTALL_PLUGIN` avant l'exécution des fonctions de désinstallation, de la façon suivante :

```
if( !defined( 'ABSPATH') && !defined('WP_UNINSTALL_PLUGIN') )
  exit();

// On supprime !
delete_option('mon_option');
```

Une fois le contrôle de sécurité effectué, il faut appeler les fonctions de désinstallation. Cela consiste à supprimer les options et également les tables de contenu s'il y en a.

Méthode 2 : utilisation du crochet de désinstallation

Le crochet fonctionne de manière un peu différente de la méthode 1. Contrairement à ce qui se passe avec la méthode 1, si on utilise le crochet, le code de l'extension sera exécuté une dernière fois avant sa suppression. Il faut donc s'assurer que le code de l'extension est bien isolé et ne risque pas de gêner la désinstallation.

Cette seconde méthode peut être très pratique pour les extensions qui ne comprennent qu'un seul fichier PHP ; cela évite de créer inutilement un dossier pour le seul fichier uninstall.php.

Pour enregistrer une fonction PHP de désinstallation, il suffit d'utiliser la fonction `register_uninstall_hook()`. Elle prend deux paramètres : le chemin complet du fichier et le nom de la fonction à exécuter.

Notez que ce crochet est appelé uniquement si le fichier uninstall.php n'existe pas ; les deux méthodes ne peuvent pas être additionnées.

```
if ( function_exists('register_uninstall_hook') )  {
  register_uninstall_hook(__FILE__, 'my_uninstall_hook');
}

function my_uninstall_hook() {
  // On supprime !
  delete_option('mon_option');
}
```

Comme vous le voyez, vous pouvez garder une compatibilité avec les anciennes versions de WordPress en utilisant la fonction `function_exists()` avant l'appel `register_uninstall_hook()`.

Initialisation de l'extension

Principe

Dans le processus d'exécution de WordPress, ce dernier va lancer dans un premier temps toutes ses bibliothèques de fonctions PHP, puis il va récupérer la liste des extensions dans une option de la base de données.

Il va ensuite faire une boucle dans le tableau des extensions et lancer une par une les extensions. Lorsqu'elles sont toutes lancées, WordPress exécute une action nommée `plugins_loaded`. Cette action permet de démarrer un traitement une fois que toutes les extensions sont chargées en mémoire.

Pour optimiser au maximum la compatibilité entre les extensions, il est fortement conseillé de lancer la logique de son extension à ce moment-là et non au chargement du fichier PHP. Cela permet par exemple à certaines extensions de se lancer avant les autres grâce au mécanisme de priorité des actions. C'est généralement le cas des extensions qui proposent d'avoir un site multilingue. Nous allons voir comment cela se passe dans le code.

Le code

Dans cet exemple, nous souhaitons ajouter un copyright à la fin du contenu des articles. Notre extension contient la fonction suivante :

```
function copyright_content( $content = '' ) {
  return $content . '<br />Copyright 2010 - Ma société';
}
```

Cette fonction est couplée à l'événement `the_content` *via* la fonction :

```
add_filter('the_content', 'copyright_content');
```

Nous allons faire en sorte de lancer l'extension au bon moment. Pour cela, il faut créer une fonction `copyright_init()` qui va contenir l'enregistrement du filtre.

Ainsi, la ligne `add_filter` est déplacée dans la fonction d'initialisation :

```
function copyright_init() {
  add_filter('the_content', 'copyright_content');
}
```

Nous couplons alors la fonction d'initialisation avec l'action `plugins_loaded` *via* la fonction `add_action()` :

```
add_action('plugins_loaded', 'copyright_init');
```

Le code final est le suivant :

```
function copyright_content( $content = '' ) {
  return $content . '<br />Copyright 2010 - Ma société';
}
```

```
function copyright_init() {
  add_filter('the_content', 'copyright_content');
}
add_action('plugins_loaded', 'copyright_init');
```

Les options de WordPress

Concept

WordPress dispose d'une table dans la base de données nommée `wp_options`. Cette dernière contient l'intégralité du paramétrage du site. Les extensions ont également la possibilité d'enregistrer leur configuration dans cette table.

Chaque option est composée de deux valeurs : son nom et sa valeur. Le nom doit être une valeur unique. Deux options ne peuvent pas avoir le même nom.

Une option peut être automatiquement chargée en mémoire lors de l'exécution de WordPress, ce qui est pratique lorsqu'on souhaite stocker une information qui s'affichera toujours (le nom du site, par exemple).

Fonctions de base

WordPress propose quatre fonctions pour gérer les options. *Via* ces fonctions, nous pouvons ajouter, mettre à jour, effacer et récupérer des options.

get_option($name)

La fonction `get_option()` permet de récupérer la valeur d'une option.

Elle prend un paramètre qui est le nom unique de l'option. Si l'option n'existe pas, la fonction retournera la valeur booléenne `false`.

Il est également possible d'utiliser un second paramètre pour préciser la valeur à retourner (autre que `false`) en cas d'absence de l'option voulue.

add_option($name, $value = '', $deprecated = '', $autoload = 'yes')

Pour ajouter une option, vous pouvez faire appel à la fonction `add_option()`. Elle peut prendre quatre paramètres, mais seul le premier est obligatoire:

- Le nom unique de l'option, qui est obligatoire.
- La valeur de l'option (par défaut, une chaîne vide).
- Le troisième paramètre n'est plus utilisé car il est déprécié.
- Le lancement automatique ou non de l'option dans WordPress. Par défaut, les options sont lancées automatiquement. Si vous ne voulez pas de ce comportement, utilisez `'no'`.

Si l'option ne contient pas de nom ou si le nom est déjà utilisé, la fonction retournera `false` et aucune modification ne sera effectuée.

update_option($option_name, $newvalue)

Pour mettre à jour une option, vous devez utiliser la fonction `update_option()`. Cette dernière prend deux paramètres :

- le nom de l'option qui va être mise à jour ;
- la nouvelle valeur de l'option.

Si l'option que vous voulez mettre à jour n'existe pas, la fonction `add_option()` sera appelée implicitement et l'option sera créée.

delete_option($name)

Pour effacer une option, il suffit d'utiliser la fonction `delete_option()` et de préciser le nom de l'option. Si l'option existe, elle sera effacée et la fonction retournera la valeur `true`, sinon elle retournera la valeur `false`.

Automatiser la création d'une interface d'administration

Concept

Afin de standardiser les pages de réglages des extensions et permettre la modification des pages de réglages incluses dans WordPress (voir Figure 10.03), l'équipe de développement a créé quatre fonctions pour gérer les sections et les champs.

Figure 10.03
Mise en page de réglages dans WordPress.

Fonctions

add_settings_section($id, $title, $callback, $page)

Cette première fonction permet d'ajouter une section dans la page de réglages. Les sections permettent de regrouper les différents champs des options dans des groupes logiques. Par exemple, vous pourrez créer une section XML-RPC et y inclure les différentes options dont vous disposez à ce sujet (activation ou autre).

La fonction prend quatre paramètres : un `id` unique qui sera utilisé sur la balise HTML générée, le titre de la section qui s'affichera dans la page de réglages, la fonction de callback qui sera appelée après le titre et avant les options. *Via* cette fonction, il sera possible d'afficher une description pour la section par exemple.

Le dernier paramètre est le nom de la page où s'affichera la section. Par exemple, dans la page de réglages écriture, le nom de la page sera `writing`. Autres noms de pages possibles : `general`, `reading`, `media`, etc.

add_settings_field($id, $title, $callback, $page, $section = 'default', $args = array())

Nous savons que nous avons la possibilité d'ajouter des sections pour regrouper les réglages, ce qui veut dire que nous pouvons ajouter des réglages. Pour cela, il existe une fonction `add_settings_field()`. Elle prend six paramètres :

- Un `id` unique, qui sera utilisé sur la balise HTML générée ; en général, il s'agit de l'identifiant de l'option.
- Le titre du réglage.
- La fonction de callback, qui permettra d'afficher un contenu pour l'option, par exemple un champ HTML de type `input`.
- La page où apparaîtra le réglage.
- La section à laquelle appartiendra le réglage.
- Un tableau PHP d'options permettant éventuellement de spécifier un label sur le titre pour améliorer l'ergonomie.

do_settings_sections($page)

Cette fonction pourra être utilisée par les extensions pour afficher une liste de sections et de réglages. Elle prend comme seul paramètre l'identifiant de la page que vous souhaitez afficher.

Dans le cadre d'une extension, il conviendra de définir un identifiant de page unique qu'il faudra utiliser avec cette fonction et les fonctions d'enregistrement de sections et d'options.

do_settings_fields($page, $section)

Cette fonction sera rarement utilisée dans les extensions. Elle permet d'afficher les options depuis une page et une section donnée. En pratique, elle est appelée implicitement par la fonction `do_settings_sections()` qui affichera les options de chaque section.

Déclarer les options dans WordPress

Concept

Dans les prochaines versions de WordPress, il sera obligatoire de déclarer explicitement les options de ses extensions pour pouvoir les utiliser.

Les développeurs de WordPress ont ajouté deux fonctions à cet effet dans WordPress 2.7, `register_setting()` et `unregister_setting()`, pour déclarer et supprimer des options. Ces fonctions ne sont que des alias vers les fonctions `add_option_update_handler()` et `remove_option_update_handler()` que vous pouviez trouver dans WordPress MU.

Ces fonctions ne sont pas encore obligatoires, mais il est souhaitable d'utiliser toutes les fonctionnalités de WordPress afin de ne pas avoir de mauvaise surprise avec vos extensions lors d'une future mise à jour de WordPress.

La déclaration présente trois principaux intérêts :

- Création de listes saines d'options.
- Appel de fonctions de callback pour formater et sécuriser les options.
- En mode multisite, les administrateurs pourront prédéfinir facilement les réglages des extensions pour l'ensemble de la plate-forme.

Fonctions

`register_setting($option_group, $option_name, $sanitize_callback = '')`

Cette première fonction permet de déclarer une option. Elle peut prendre trois paramètres : le nom du groupe de l'option (par exemple, il peut s'agir d'un identifiant commun à une extension), l'identifiant de l'option, et optionnellement la fonction de callback à appeler pour formater la valeur. Par exemple, si nous savons que la valeur retournée doit être un entier, nous passerons comme troisième argument la valeur `intval` pour appeler la fonction PHP du même nom et ainsi sécuriser la valeur.

`unregister_setting($option_group, $option_name, $sanitize_callback = '')`

Cette fonction permet de supprimer une déclaration. Pour cela, il faut fournir les trois mêmes paramètres que la déclaration.

`settings_fields($option_group)`

Cette fonction est à utiliser sur la page des options de l'extension, dans l'interface d'administration ; elle permet d'afficher deux champs d'entrée pour spécifier le groupe d'options dans lequel vous travaillez, l'action que vous réalisez (mise à jour) et le "nonce" de sécurité pour la page (*cf.* la section dédiée aux "nonces"). Elle prend comme seul paramètre le nom du groupe d'options.

Bonnes pratiques

En règle générale, lors du développement d'une extension, il faut être économe en options. Il est préférable de stocker vos réglages dans une SEULE option, *via* un tableau PHP, plutôt qu'utiliser une option par réglage.

Par exemple, si votre extension contient dix paramètres, vous obtiendrez de meilleures performances en stockant les dix valeurs dans un tableau, lui-même stocké dans une option, qu'avec les dix valeurs dans dix options différentes.

De plus, l'utilisation des tableaux est facilitée avec les fonctions vues ci-dessus. Ces dernières gèrent dynamiquement la sérialisation et la désérialisation du tableau pour permettre le stockage dans la base de données.

Attention cependant à ne pas stocker trop de valeurs dans un seul tableau. Il risque de consommer plus de mémoire, et le gain de performances pourrait poser problème sur les hébergements avec peu de mémoire allouée à PHP. Par exemple, il n'est pas interdit de stocker les réglages nécessaires uniquement à l'administration dans une option, et les autres réglages dans une seconde option.

Enfin, les réglages des widgets sont généralement stockés dans une option différente de l'extension.

Les permissions et les rôles

Concept

L'administrateur d'un site WordPress peut créer d'autres comptes utilisateur et leur assigner différents droits par le biais des rôles. Ces rôles définissent ce que l'utilisateur peut faire : écrire un article, modifier une page, supprimer une catégorie, installer un thème...

Principes

WordPress contient par défaut cinq rôles :

- administrateur ;
- éditeur ;
- auteur ;
- contributeur ;
- abonné.

À ces rôles s'ajoute, depuis la version 3.0, le statut de "super-admin", qui est un privilège accessible à partir du moment où WordPress fonctionne en mode multisite. Le super-admin, qui est l'administrateur du réseau de site, dispose de droits supplémentaires relatifs à la gestion de l'ensemble du réseau.

Chaque rôle dispose de permissions (*capabilities* en anglais). Plus le rôle est important, plus les permissions sont étendues. Par exemple, par défaut, seul le rôle administrateur a le droit de modifier les réglages du site, car lui seul possède la permission `manage_options`.

Ces permissions peuvent être visionnées et modifiées *via* l'extension Role Manager (voir Figure 10.04). Cette dernière permet de configurer totalement les permissions de chaque rôle.

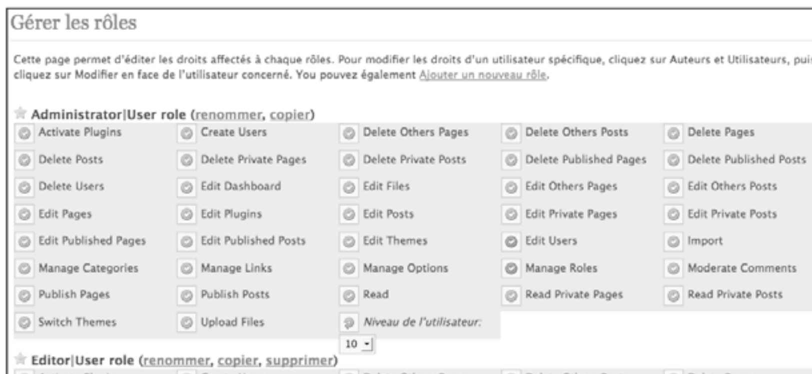


Figure 10.04
Les permissions par rôle dans l'extension Role Manager.

Vous pouvez également créer de nouveaux rôles avec les permissions souhaitées *via* cette extension. Ces actions peuvent être réalisées aussi depuis le code de votre extension *via* l'API de Rôles de WordPress.

Dans cette section, nous n'allons pas décortiquer l'intégralité des fonctions, mais uniquement les plus utilisées. N'hésitez pas à lire le code source de WordPress pour découvrir toutes les méthodes.

Les niveaux de WordPress – rappel

Avant de détailler les fonctions de l'API de Rôles, faisons un petit rappel sur l'ancien fonctionnement des utilisateurs de WordPress.

Avec la version 2.0, les utilisateurs ne travaillaient pas avec des permissions, mais uniquement sur des niveaux (levels en anglais). Plus le niveau était grand, plus l'utilisateur possédait de droits. Par exemple, le rôle administrateur correspondait au niveau 10.

Ce système, limité techniquement, empêchait une configuration fine des permissions. Par exemple, une extension pouvait difficilement créer un rôle et spécifier précisément le droit pour chacune de ces fonctionnalités.

Aujourd'hui, il est toujours possible d'utiliser les niveaux dans WordPress pour des raisons de compatibilité avec les anciennes extensions ; par exemple, les menus peuvent encore recevoir un niveau. Néanmoins, cette façon de travailler est complètement déconseillée.

Classes PHP

WordPress possède trois classes pour gérer les rôles, permissions et utilisateurs. Vous les retrouvez ci-dessous.

WP_Roles

Cette classe gère les rôles dans WordPress.

Vous pouvez :

- ajouter un rôle ;
- supprimer un rôle ;
- récupérer un rôle ;
- ajouter une permission à un rôle ;
- supprimer une permission à un rôle.

WP_Role

Cette classe est présente à des fins de rétrocompatibilité pour les anciennes extensions. Elle ne doit pas être utilisée par les nouvelles extensions.

WP_User

Cette classe gère l'interaction entre les utilisateurs et les rôles.

Vous pouvez :

- récupérer le rôle d'un utilisateur ;
- attribuer un rôle à un utilisateur ;
- supprimer un rôle à un utilisateur ;
- ajouter un rôle à un utilisateur ;
- ajouter une permission à un utilisateur ;
- supprimer une permission à un utilisateur ;
- vérifier que l'utilisateur possède une permission spécifique ;
- traduire le niveau (level) d'un utilisateur en permission.

Fonctions d'aide

Afin de simplifier la vie des développeurs et leur éviter de trop instancier ces classes (ce qui complexifie le code et peut faire fuir les développeurs novices), quatre fonctions PHP classiques permettent de travailler avec les rôles.

current_user_can($capability)

Cette première fonction permet de vérifier que l'utilisateur connecté possède bien la permission ou le rôle indiqué.

Cela peut être très utile pour améliorer la sécurité d'une extension.

Par exemple, lors de l'envoi des données d'un formulaire HTML, vous placez généralement le contrôle et l'enregistrement des données avant l'affichage afin de pouvoir faire une redirection par exemple.

Dans ce cas précis, vous ne profitez pas du contrôle de sécurité de WordPress. Il faut donc sécuriser manuellement votre fonction ; pour cela `current_user_can()` est très simple à utiliser.

Cette fonction prend un seul paramètre : le nom de la permission ou du rôle que vous souhaitez tester sur l'utilisateur connecté. Elle retourne la valeur `true` si l'utilisateur possède la permission/le rôle, sinon elle retourne `false`.

get_role($role), add_role($role, $display_name, $capabilities = array()), remove_role($role)

Ces trois fonctions sont des raccourcis pour l'utilisation de la classe `WP_Roles`.

Elles instancient automatiquement la classe et appellent les méthodes du même nom de la classe.

Ces fonctions sont présentes pour permettre aux développeurs débutants de ne pas avoir à se soucier des concepts de classes, d'objets et d'instanciations.

get_role($role)

La fonction `get_role()` permet de récupérer les permissions d'un rôle. Elle prend en seul paramètre le nom du rôle.

add_role($role, $display_name, $capabilities = array())

La fonction `add_role()` ajoute un rôle dans WordPress. Elle prend trois paramètres :

- le nom administratif du rôle ;
- le nom public du rôle ;
- le tableau des permissions associées au rôle.

remove_role($role)

Enfin, la fonction `remove_role()` permet la suppression d'un rôle de WordPress ; elle prend comme seul paramètre le nom du rôle.

is_super_admin($user_id = false)

Cette nouvelle fonction, utile principalement en mode multisite, aide simplement à déterminer si un utilisateur est super-admin ou non. Son unique paramètre permet de préciser l'ID de l'utilisateur, mais il est optionnel : sans valeur, c'est le statut de l'utilisateur courant qui est vérifié.

Exemples

Nous allons voir comment ajouter une permission à un rôle.

Ici nous souhaitons ajouter la permission `gestion_monplugin` au rôle administrateur de WordPress. Cette permission permettra de gérer les données d'une extension de notre création.

Le code que vous avez ci-après ne doit pas être exécuté à chaque chargement de l'extension. En règle générale, vous le retrouvez lors de l'activation de l'extension :

```
// On récupère les données du rôle administrateur
$role = get_role('administrator');

// On vérifie que le rôle existe et qu'il ne possède déjà pas la permission
if( $role != null && !$role->has_cap('gestion_monplugin') ) {
  // Si la permission n'existe pas, on ajoute la permission au rôle.
  $role->add_cap('gestion_monplugin');
}
```

Bonnes pratiques

Aucun développeur n'est obligé de créer des permissions spécifiques pour son extension. Dans un certain nombre de cas, les permissions par défaut suffisent.

Pour sécuriser la partie réglages de son extension, il suffit de contrôler la permission `manage_options`.

Si l'extension gère un nouvel aspect des mots-clefs, la permission liée aux catégories et aux mots-clefs de WordPress `manage_categories` fera également l'affaire.

La liste des rôles et des permissions associées est présente sur le Codex de WordPress (**http://codex.wordpress.org/Roles_and_Capabilities**). Cette page peut être très utile durant le développement.

Internationalisation de WordPress

Concept

Comme vous avez pu le lire au Chapitre 1, WordPress est un logiciel développé en anglais. Pour permettre la plus large diffusion possible du logiciel, les développeurs ont internationalisé WordPress. Cela veut dire qu'il peut être traduit très facilement sans modifier les fichiers PHP. Il utilise pour cela le format de fichier de la bibliothèque open-source gettext.

Grâce à ce format, il est possible de répertorier toutes les chaînes de textes dans un fichier PO. Ce fichier PO est éditable *via* un éditeur de texte ou des logiciels spécialisés, comme poEdit. Il contient les informations suivantes :

- le nom du fichier où se situe la chaîne, ainsi que la ligne ;
- la chaîne dans la langue originale ;
- la chaîne traduite.

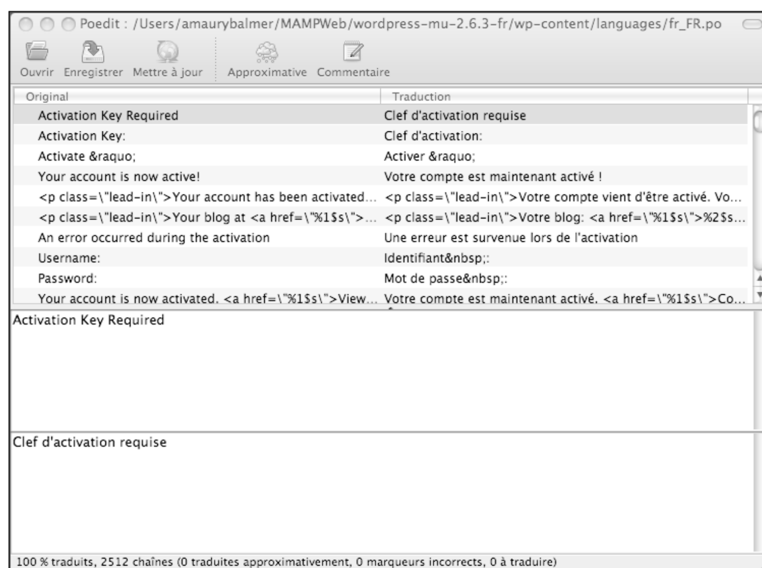
Cela donne :

```
#: wp-admin/admin-header.php:73
msgid "Visit Site"
msgstr "Aller sur le site"
```

De nos jours, les fichiers PO ne sont plus modifiés *via* des éditeurs de texte. Il existe un excellent logiciel disponible sous Windows, Linux et Mac, appelé poEdit (voir Figure 10.05). Comme son nom l'indique, c'est un éditeur de fichier PO ; il est libre, gratuit, disponible dans de nombreuses langues et dispose de toutes les fonctionnalités nécessaires à l'internationalisation *via* le format de fichier gettext.

Figure 10.05

Fenêtre de l'application poEdit.



Lors de l'enregistrement du fichier PO avec poEdit, le logiciel crée un fichier avec une extension .mo. Un fichier MO est un binaire ; c'est une compilation du fichier PO qui a pour but d'être plus légère en taille et d'avoir de meilleures performances. Les fichiers MO sont les seuls fichiers lisibles par la librairie gettext. Un fichier MO n'est pas éditable ; généralement, les fichiers PO et MO sont toujours diffusés ensemble, afin de faciliter la correction de la traduction.

Implémentation dans WordPress

WordPress possède une classe qui émule le fonctionnement de gettext. Cette classe provoque une baisse sensible des performances, mais elle offre plus de souplesse sur le positionnement des fichiers PO/MO dans le système de fichiers. Dans le pack français de WordPress, vous pouvez trouver la traduction française dans le dossier wp-content/languages.

Dans les versions les plus anciennes du pack, la traduction peut également se situer dans le dossier wp-includes/languages. Cet emplacement est cependant obsolète, car régulièrement effacé lors des mises à jour...

Pour activer une langue dans WordPress, il faut modifier le fichier de configuration wp-config.php et éditer la valeur de la constante `WPLANG`.

Pour la langue française, vous utiliserez :

```
define ('WPLANG', 'fr_FR');
```

Fonctions

Au fil des versions et des besoins, les développeurs de WordPress ont développé un bon nombre de fonctions dédiées à l'internationalisation du logiciel. Par un souci de concision, certaines ne seront abordées que superficiellement.

get_locale()

Cette fonction retourne le code de la langue chargée dans WordPress. Si vous utilisez la version française de WordPress, la valeur retournée sera `fr_FR`.

get_available_languages($dir = null)

Cette fonction retourne un tableau PHP contenant la liste des langages disponibles, en se fondant sur les fichiers .mo présents dans un dossier donné (par défaut, celui de la constante WP_LANG_DIR, soit normalement wp-content/languages).

__($text, $domain = 'default')

Cette fonction permet de retourner une chaîne de caractères traduite. Elle prend deux paramètres : le premier est la chaîne de caractères à traduire. Généralement, dans les projets libres, c'est une chaîne de langue anglaise.

Le second paramètre est le domaine de traduction. Grâce à ce domaine, il est possible de charger un fichier MO spécifique par domaine. Si aucun domaine n'est précisé, le domaine utilisé sera celui par défaut. Le fichier de traduction de WordPress traduit uniquement les chaînes avec le domaine par défaut.

Dans le cadre d'une extension, il est nécessaire de créer son propre domaine et, du coup, de charger le fichier MO spécifique à l'extension (ce qui sous-entend de créer entre les deux un fichier PO).

_e($text, $domain = 'default')

Cette fonction fait exactement la même chose que la fonction `__()`, à la différence qu'au lieu de retourner la valeur traduite, la fonction `_e()` l'affiche *via* la fonction PHP `echo()`.

_x($text, $domain = 'default')

Cette fonction est très semblable à la fonction `__()`. Elle supporte néanmoins la notion de contexte dans la chaîne de caractères à traduire. En effet, pour faciliter la traduction de

certaines chaînes, les développeurs ont la possibilité d'apporter quelques précisions sur le contexte. Avant WordPress 2.8, cette version était appelée `_c()`.

Le contexte est séparé de la chaîne *via* le caractère `|`.

Exemple :

```
_c("Approved|plural");
```

Avec le commentaire, vous savez que la chaîne de caractères est utilisée dans un contexte pluriel. En anglais, il n'y a pas de distinction entre le singulier et le pluriel, mais dans d'autres langues comme le français, ce n'est pas la même chose.

Le commentaire ne doit pas être traduit.

_n($single, $plural, $number, $domain = 'default')

Cette fonction est un peu plus compliquée que les précédentes. Elle permet de gérer la traduction d'une chaîne de caractères dans un contexte singulier et pluriel. Avant WordPress 2.8, cette version était appelée `__ngettext()`.

Exemple dans le fichier PO :

```
#: wp-admin/edit-comments.php:80
#, php-format
msgid "%s comment deleted"
msgid_plural "%s comments deleted"
msgstr[0] "%s commentaire supprimé"
msgstr[1] "%s commentaires supprimés"
```

Cette fonction prend quatre paramètres :

- La chaîne de caractères au singulier.
- La chaîne de caractères au pluriel.
- Le chiffre ou le nombre de la situation. Dans l'exemple ci-après, c'est le nombre de commentaires.
- Le domaine de traduction.

Le fichier PO génère ici deux possibilités : singulier et pluriel.



Certaines langues ont trois possibilités : zéro, un et pluriel. Ce paramètre est défini dans l'en-tête du fichier PO.

_nx($single, $plural, $number, $context, $domain = 'default')

Comme son nom le laisse deviner, cette fonction combine les comportements de `_n()` et `_x()` : elle permet de préciser à la fois le contexte singulier et pluriel, et d'indiquer le domaine auquel la traduction s'applique (si besoin est).

esc_attr_*() et esc_html_*()

Les fonctions `__()`, `_e()` et `_x()` disposent toutes les trois de variantes spécialisées, nommées sur le même modèle :

- `esc_attr_e()`
- `esc_attr_x()`
- `esc_attr__()`
- `esc_html_e()`
- `esc_html_x()`
- `esc_html__()`

Leurs dénominations proches nous permettent de les expliquer par groupe :

- "esc" signifie que ces fonctions renvoient un contenu "échappé", c'est-à-dire que certains caractères spéciaux (esperluettes, guillemets, apostrophes...) ont été remplacés par des entités HTML.
- "attr" signifie que le contenu est échappé pour être inclus dans un attribut de balise HTML, tandis que "html" indique que le contenu peut être utilisé tel quel au sein de code HTML.

Ensemble, ces six fonctions permettent de gérer les traductions au sein d'un fichier HTML de manière élégante et sûre, que ce soit le contenu d'un attribut `alt=""` ou du texte normal.

load_textdomain($domain, $mofile)

Cette fonction permet de charger un fichier MO en mémoire et de l'associer à un domaine de traduction.

Le premier paramètre est le domaine de traduction, et le second est le chemin complet vers le fichier de traduction MO.

De son côté, `unload_textdomain($domain)` permet de supprimer de la mémoire les traductions chargées pour un domaine donné.

is_textdomain_loaded($domain, $path = false)

Cette fonction compagne de la précédente permet de vérifier si des traductions sont disponibles pour le domaine précisé, en renvoyant true si c'est le cas, false le cas échéant.

load_plugin_textdomain($domain, $abs_rel_path = false, $plugin_rel_path = false)

Cette fonction a comme objectif de simplifier le lancement de fichier MO pour les extensions. Par défaut, la fonction tente de charger le fichier MO depuis le répertoire wp-content/plugins.

En voici un exemple : une extension de calendrier utilise le domaine `moncalendrier` et le site installé est traduit en français (code langue : `fr_FR`).

La fonction tentera de charger le fichier wp-content/plugins/moncalendrier-fr_FR.mo.

Le deuxième paramètre permet de spécifier le chemin du fichier de traduction MO. WordPress ajoute automatiquement la constante ABSPATH en préfixe à la valeur passée en paramètre.

Le troisième paramètre permet également de spécifier le chemin vers la traduction ; il préfixe la valeur passée en paramètre de la constante WP_PLUGIN_DIR.

Pour charger la traduction contenue dans le dossier languages de l'extension Mon Calendrier, nous avons cette possibilité :

```
load_plugin_textdomain($domain, 'wp-content/plugins/moncalendrier/languages',
false);
```

Ou celle-là :

```
load_plugin_textdomain($domain, false, 'moncalendrier/languages');
```

La dernière méthode est recommandée par les développeurs de WordPress.

Cette fonction dispose d'un équivalent spécialisé pour les extensions placées dans le dossier mu-plugins : `load_muplugin_textdomain($domain, $mu_plugin_rel_path = '')`.

load_theme_textdomain($domain, $path = false)

Cette dernière fonction permet de charger facilement les traductions dans les thèmes WordPress.

Elle prend comme premier paramètre le domaine de traduction. La fonction essaiera uniquement de charger le fichier MO du thème activé.

Le nom et la localisation des fichiers MO pour les thèmes sont très simples, il suffit de nommer le fichier MO par le code de la langue et de le positionner à la racine du thème.

Par exemple, pour le thème K2, il faut placer la traduction française à l'emplacement suivant : wp-content/themes/k2/fr_FR.mo.

Cette fonction dispose d'un équivalent spécialisé pour les thèmes enfants : `load_child_theme_textdomain($domain, $path = false)`.

Bonnes pratiques

Une erreur basique revient fréquemment dans l'internationalisation des extensions et thèmes. Cette erreur survient lorsqu'on travaille avec des phrases comprenant des variables susceptibles de changer.

L'erreur est la suivante :

```
__('Replace ').$a.__(' with ').$b;
```

Ici, vous voyez que la phrase contient deux variables susceptibles de changer : `$a` et `$b`. Le problème, c'est que le découpage fonctionne dans la langue anglaise. Mais fonctionne-t-il avec toutes les langues ?

Dans certaines langues, il est probable que la valeur du `$b` doive être positionnée avant le `$a`. Pour simplifier la traduction et la rendre la plus compatible possible, il est intéressant d'utiliser les chaînes formatées de PHP.

Vous remplacez ainsi la chaîne de l'exemple précédent par :

```
sprintf(__('Replace %1$s with %2$s'), $a, $b);
```

Ainsi, vous pouvez traduire comme bon vous semble la phrase, avec la possibilité de mettre la variable `$b` avant la variable `$a`.

Notez qu'en règle générale la construction des phrases en anglais et en français est sensiblement la même.

Cette erreur n'est donc pas dramatique pour la traduction française ; il n'empêche qu'une meilleure internationalisation de votre extension ou thème ne pourra qu'augmenter sa diffusion à l'échelle mondiale.

WP_Http : l'API HTTP

Concept

Au fil des évolutions, WordPress est amené de plus en plus souvent à effectuer des appels HTTP pour obtenir des données et des informations sur des sites extérieurs.

Nous retrouvons ces appels pour différentes fonctionnalités :

- le contrôle de version de WordPress et des extensions ;
- la mise à jour de WordPress et des extensions ;
- les pings effectués par les extensions tierces comme Google Sitemap Generator, etc.

Malheureusement, jusqu'à la version 2.7 de WordPress, il n'existait pas d'API pour standardiser les appels HTTP de WordPress et ses extensions. De ce fait, il était possible de trouver dans WordPress différentes méthodes pour effectuer ces appels.

Afin de standardiser ces appels et par la même occasion améliorer la compatibilité et les performances de WordPress, l'équipe de WordPress a développé une nouvelle classe HTTP.

Notez qu'il existe une classe PHP nommée `Snoopy`, présente dans WordPress depuis la version 1.5. Elle a pour but d'émuler un navigateur web ; de ce fait elle pourrait répondre au besoin des appels HTTP. Malheureusement, cette classe est très ancienne, et elle ne correspond pas au besoin d'amélioration des performances et des compatibilités, d'autant plus qu'elle ne supporte que deux technologies sur les cinq existantes (elles seront détaillées ci-après).

Rappel : les différents types de requêtes HTTP

Il existe différents types de requêtes HTTP ; ces types sont définis par une norme RFC qui en est à sa version 1.1. Les serveurs HTTP implémentent au mieux ses fonctionnalités en natif ou *via* des extensions.

Ici, nous présenterons uniquement les trois types de requêtes les plus utilisées ; de ce fait elles sont gérées nativement dans tous les serveurs HTTP modernes que vous trouvez sur le marché.

Vous trouverez plus d'informations sur la norme HTTP 1.0 et 1.1 sur Wikipédia : **http://fr.wikipedia.org/wiki/Hypertext_Transfer_Protocol**.

- **`GET`.** C'est la méthode la plus courante pour demander une ressource. Une requête `GET` est sans effet sur la ressource, il doit être possible de répéter la requête sans effet.
- **`HEAD`.** Cette méthode ne veut que des informations sur la ressource, sans demander la ressource elle-même.
- **`POST`.** Cette méthode doit être utilisée pour ajouter une nouvelle ressource (un message sur un forum ou un article dans un site). Néanmoins, en pratique, il n'est pas rare de devoir effectuer un appel de type `POST` pour simplement demander une ressource sans pour autant la modifier.

Technologies

Il existe cinq technologies différentes dans PHP pour faire des appels sur des ressources externes. Le but ici n'est certainement pas de comparer les plus et les moins de chaque technologie, mais d'expliquer le mode de fonctionnement de la classe.

La classe `WP_Http` est capable d'utiliser ces cinq technologies. Le choix de la technologie sera défini par ordre de performances. On essaiera d'utiliser la technologie la plus performante avant de passer à la suivante, ce qui donne cet ordre-ci :

- HTTP (classe propre à PHP 5.2) ;
- cURL ;
- Streams ;
- Fopen ;
- Fsockopen.

Si la technologie n'est pas disponible, parce que la version de PHP est plus ancienne ou parce que l'hébergeur l'a désactivée explicitement, WordPress passera automatiquement à la suivante.

En théorie, cette classe est compatible avec la quasi-totalité des hébergeurs et garantit les meilleures performances possibles pour votre site.

Fonctions

_wp_HTTP_get_object()

Cette fonction permet de retourner un objet qui représente une instanciation de la classe `WP_Http` de WordPress. Avec cet objet, il sera possible d'utiliser les méthodes de la classe.

wp_remote_request($url, $args = array())

Cette fonction permet de faire une requête HTTP complexe. Le premier paramètre est l'adresse de la ressource, et le second paramètre est un tableau PHP assez complexe contenant les informations de l'en-tête HTTP ; vous pouvez ainsi définir la méthode, le délai avant expiration, le suivi ou non des redirections, la version du protocole HTTP, l'agent utilisateur employé… Le tableau complet est disponible depuis le fichier wp-includes/HTTP.php contenant la classe `WP_Http`.

wp_remote_get($url, $args = array())

Cette fonction est un alias vers la fonction `wp_remote_request()`. Elle spécifie juste que la méthode employée sera `GET`. Les deux paramètres sont donc les mêmes, sauf que la méthode HTTP ne peut pas être modifiée depuis le second paramètre.

wp_remote_post($url, $args = array())

Cette fonction est semblable à `wp_remote_get()`, à la différence que la méthode employée sera `POST`.

wp_remote_head($url, $args = array())

Cette fonction est semblable à `wp_remote_post()`, à la différence que la méthode employée sera `HEAD`.

wp_remote_retrieve_headers(&$response)

Cette fonction permet de récupérer le tableau de l'en-tête HTTP. Elle prend comme seul paramètre la réponse HTTP que l'on aura récupérée *via* une des fonctions présentées auparavant.

wp_remote_retrieve_header(&$response, $header)

Cette fonction est un peu plus précise que `wp_remote_retrieve_headers()` ; elle permet de récupérer une valeur précise de l'en-tête HTTP. Pour cela, il faut passer en plus du premier paramètre (la réponse HTTP) le champ de l'en-tête à récupérer, par exemple `code`, pour retrouver le code HTTP de la ressource.

wp_remote_retrieve_response_code(&$response)

Cette fonction permet de retourner le code HTTP de réponse de la requête. Elle utilise en fait la fonction `wp_remote_retrieve_header()` et précise comme second paramètre l'identifiant pour le code HTTP, c'est-à-dire `code`.

wp_remote_retrieve_response_message(&$response)

Tout comme `wp_remote_retrieve_response_code()`, cette fonction est un alias de `wp_remote_retrieve_header()`. Le deuxième paramètre passé sera `message`. Il permet de récupérer cette information de la réponse HTTP.

wp_remote_retrieve_body(&$response)

Enfin, cette dernière fonction permet de récupérer le contenu de la réponse HTTP. Elle prend comme seul paramètre la réponse HTTP.

Par exemple, si vous appelez une page comme http://google.fr, vous récupérerez une longue chaîne de caractères contenant tout le code HTML de la page d'accueil de Google.

L'API shortcode

Concept

Une des nouveautés de WordPress 2.5 est l'ajout de galeries photo dans le logiciel.

Pour permettre une insertion facile des galeries (voir Figure 10.06) et standardiser le remplacement des marqueurs de type BBcode dans le contenu des articles, les développeurs ont créé une API appelée shortcode.

Figure 10.06
Un marqueur shortcode dans l'éditeur visuel.



Avant cette API, chaque développeur remplaçait les marqueurs avec sa propre méthode, et cela pouvait provoquer des bogues et des problèmes de performances.

C'est le cas, par exemple, des extensions de formulaires de contact et d'insertion de lecteur vidéo.

Cette API permet le remplacement de marqueur de type BBcode, comme `[gallery]`, par un code HTML spécifique.

Ainsi, cette API permet aux développeurs de ne plus se compliquer l'existence avec les expressions régulières.

Utilisation

L'exemple le plus basique est celui de l'insertion des galeries. Il suffit d'ajouter le shortcode

```
[gallery]
```

dans le contenu de l'article pour qu'il soit remplacé par la galerie d'images WordPress.

Vous pouvez également spécifier des attributs dans le marqueur shortcode, comme ceci :

```
[gallery id="123" size="medium"]
```

Fonctions

Nous avons une série de fonctions à notre disposition, dont trois pour l'enregistrement et la suppression des shortcodes, et une dernière pour l'extraction des valeurs des attributs.

Les shortcodes fonctionnent sur le même principe que les filtres. En réalité, un shortcode n'est qu'un filtre plus évolué et subordonné à une tâche spécifique.

Comme les filtres, ils doivent être enregistrés afin d'être exécutés au bon moment par WordPress. Et, comme les filtres, un shortcode peut être ajouté ou supprimé.

add_shortcode($tag, $func)

Cette première fonction permet d'enregistrer un nouveau shortcode. Elle prend deux paramètres : le premier est le nom du marqueur du shortcode, le second est le nom de la fonction à exécuter pour remplacer le marqueur par le code HTML voulu.

Par exemple, si nous souhaitons créer le marqueur suivant :

```
[deezer id="123"]
```

Le nom du marqueur est Deezer ; nous allons coupler ce marqueur avec la fonction `deezer_shortcode()`.

L'appel de la fonction `add_shortcode()` sera :

```
add_shortcode('deezer', 'deezer_shortcode');
```

remove_shortcode($tag)

Comme son nom l'indique, cette deuxième fonction permet de supprimer un shortcode dans WordPress.

Par exemple, si vous ne souhaitez pas utiliser le shortcode des galeries de WordPress mais employer votre ancienne extension qui fonctionne avec le même shortcode, il vous suffira de choisir la fonction `remove_shortcode()` ; cette fonction prend comme seul paramètre le nom du marqueur.

Pour supprimer les galeries de WordPress, exécutez la fonction :

```
remove_shortcode('gallery');
```

remove_all_shortcodes()

Dans certains cas, il peut être nécessaire de supprimer tous les shortcodes enregistrés dans WordPress. Pour cela, il existe la fonction `remove_all_shortcodes()`.

Elle ne prend aucun paramètre ; il suffit de l'exécuter lors de l'action `init` de WordPress pour supprimer tous les shortcodes enregistrés par WordPress et les extensions.

shortcode_atts($pairs, $atts)

Contrairement aux fonctions précédentes, `shortcode_atts()` s'utilise uniquement dans la fonction appelée par le shortcode – le deuxième paramètre de la fonction `add_shortcode()`.

Cette fonction permet d'extraire les valeurs des attributs de votre shortcode. Par exemple pour le shortcode :

```
[deezer id="123"]
```

...vous pourrez récupérer la valeur 123 de l'attribut `id`. Le premier paramètre de la fonction doit être un tableau à deux dimensions contenant le nom de l'attribut et sa valeur par défaut si elle existe.

Le second paramètre est le tableau des attributs au format brut et donc peu exploitable. Généralement, la fonction `shortcode_atts()` est couplée avec la fonction PHP `extract()`. Voir l'exemple ci-après.

Exemple

Nous allons permettre le remplacement d'un shortcode par deux chaînes de texte, mais le code HTML de remplacement ne dépend que de vous… Vous pouvez très bien mettre un lecteur Flash par exemple.

Le shortcode utilisable depuis le contenu des articles est :

```
[deezer id="123" title="Texte alternatif"]
```

Voici le code de l'extension (n'oubliez pas de lire les commentaires pour comprendre la suite des événements) :

```
// On enregistre le shortcode dans WordPress.
add_shortcode('deezer', 'deezer_player');

// Ci-après la fonction appelée par le shortcode pour remplacer
function deezer_player( $atts = '' ) {
  // On récupère les valeurs autorisées par mon shortcode.
  extract(shortcode_atts(array('id' => '','title' => ''), $atts));

  // Je vérifie que la valeur du shortcode est bien un entier.
   // Si ça n'est pas le cas, je retourne une chaîne vide.
  $id = (int) $id;
  if ( $id == 0 ) {
    return '';
  }

  // On remplace le shortcode par du texte.
   // En pratique, cela peut être remplacé par un lecteur flash...
  $output = 'Vidéo ID : '. $id;
  // On peut faire de même avec le titre.
  $output .= 'Vidéo Title : '. $title;
  // On n'oublie pas de renvoyer la valeur de remplacement
  return $output;
}
```

Les menus de WordPress

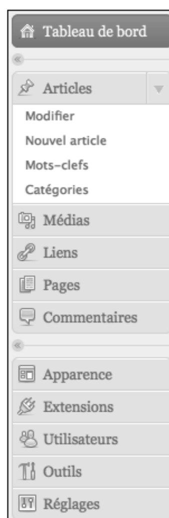
Concept

L'interface d'administration de WordPress a beaucoup évolué entre les versions 2.3 et 2.6, changeant plus ou moins radicalement de style par deux fois en un an : WordPress 2.5 tentera de consolider l'administration en place depuis la version 1.0, tandis que WordPress 2.7 se verra offrir une toute nouvelle interface, beaucoup plus à même de suivre les évolutions à venir du CMS. C'est cette dernière évolution qui est toujours en place aujourd'hui.

Malgré ces nombreuses incarnations, cette interface a toujours été pensée, dès sa conception, pour accueillir des extensions. Pour cela, l'équipe de développement met à disposition différentes fonctions pour ajouter des menus dans WordPress (voir Figure 10.07).

Ces menus peuvent être insérés dans toutes les rubriques de WordPress, que ce soit Tableau de bord, Articles, Médias, Apparence, Extensions, Réglages, etc. Désormais, la seule difficulté pour le développeur est de savoir dans quel menu il place la page de son extension. Nous aborderons cette question dans la section des bonnes pratiques.

Figure 10.07
Le menu à partir de WordPress 2.7.



Fonctions

L'API des menus contient cinq fonctions. Les deux premières que nous allons voir sont les plus importantes, les trois dernières n'étant que des fonctions simplifiées exploitant directement les deux premières fonctions.

add_menu_page($page_title, $menu_title, $capability, $menu_slug, $function = '', $icon_url = '', $position = NULL)

Cette fonction permet de créer des menus de premier niveau. Le premier niveau dans WordPress, c'est par exemple Articles ou Outils.

```
add_menu_page('Titre de la page', 'Titre de la page dans le menu', 'manage_
options', 'mon-menu-de-premier-niveau', 'fonction_de_la_page');
```

Cette fonction prend sept paramètres :

- le titre de la page, celui qu'on retrouve dans la balise HTML `title` ;
- le titre de la page que l'on retrouve dans le menu ;
- la capacité minimale de l'utilisateur (*capability* en anglais).
- l'identifiant unique de ce menu, ou le chemin complet vers le fichier PHP de la page d'administration, généré facilement avec la constante PHP `__FILE__` ;
- la fonction PHP à exécuter pour la génération de votre page ;
- l'adresse web de l'icône à utiliser pour ce menu ;
- la position de ce menu parmi les autres menus.

Les deux derniers paramètres sont facultatifs. Si aucune fonction n'est précisée, WordPress partira du principe que le fichier indiqué génère automatiquement l'écran d'administration.

add_submenu_page($parent_slug, $page_title, $menu_title, $capability, $menu_slug, $function = '')

Cette deuxième fonction est tout aussi importante que la précédente ; elle permet de créer des sous-menus. Par exemple, la page Ajouter contenue dans le menu de premier niveau Articles est un sous-menu par défaut de WordPress.

```
add_submenu_page(__FILE__, 'Titre de la page', 'Titre de la page dans le menu',
8, __FILE__, 'fonction_de_la_page');
```

Au niveau des paramètres, c'est quasi la même chose que `add_menu_page()`, sauf qu'il y a un argument en plus. Il est placé en première position ; ce paramètre spécifie la page parente du sous-menu. Vous avez deux possibilités :

- Soit vous ajoutez un sous-menu à un menu existant de WordPress, auquel cas il suffit d'entrer le nom du fichier PHP de WordPress. Par exemple, pour ajouter une page dans le menu Articles, il suffit de passer en paramètre 'edit.php'.
- Soit, dans le cas du sous-menu d'un menu ajouté *via* une extension, il suffit de passer le chemin entier vers le fichier PHP de l'extension (avec la constante PHP `__FILE__` par exemple).

Depuis WordPress 2.8, il est également possible d'utiliser l'identifiant (ou "slug") du menu supérieur, plutôt que son nom de fichier.

add_options_page($page_title, $menu_title, $capability, $menu_slug, $function = '')

Cette fonction est un raccourci pour ajouter un sous-menu du menu Réglages. Elle se contente de préciser le fichier parent à la fonction `add_submenu_page()`.

```
add_options_page(page_title, menu_title, capability, file, [function]);
```

Les paramètres sont les mêmes que la fonction `add_menu_page()`.

add_management_page($page_title, $menu_title, $capability, $menu_slug, $function = '')

Cette fonction permet d'ajouter un sous-menu dans le menu Outils.

```
add_management_page(page_title, menu_title, capability, file, [function]);
```

Elle dispose des mêmes caractéristiques que `add_options_page()`.

add_theme_page($page_title, $menu_title, $capability, $menu_slug, $function = '')

Cette fonction permet d'ajouter un sous-menu dans le menu Apparence.

```
add_theme_page( page_title, menu_title, capability, file, [function]);
```

Elle dispose des mêmes caractéristiques que `add_options_page()`.

Bonnes pratiques

Depuis la version 2.7, WordPress dispose d'une interface dont le menu se trouve à gauche de l'écran, plutôt qu'en haut de celui-ci comme c'était le cas depuis le début. Ce changement fondamental d'interface a permis, entre autres choses, de ne plus voir les menus de l'administration rendus inutilisables par un trop-plein de pages d'extensions. En effet, il suffisait d'une dizaine d'extensions pour que les menus "explosent" en largeur et cassent la mise en page de l'interface.

Grâce à la nouvelle disposition horizontale, il est désormais possible d'avoir un grand nombre d'extensions avec leurs pages, sans amener de la confusion dans l'interface. Il est même possible pour une extension de créer son propre menu de premier niveau.

L'objectif premier est de ne pas surcharger l'interface de sous-menus, afin de garder l'ensemble lisible pour l'utilisateur. La simplicité veut que les extensions placent leurs pages dans le menu Réglages (Options en anglais) – avec toujours la possibilité de placer une page dans un autre menu (par exemple Articles) si cela se justifie. L'utilisation d'un menu de premier niveau est à réserver aux extensions qui introduisent des concepts totalement nouveaux dans l'interface de WordPress, concept qui ne peut être logiquement pris en compte dans les menus existants.

Une autre solution existe par ailleurs : créer une page principale pour son extension, et mettre le menu de l'extension directement sur cette page, plutôt que charger le menu principal ou créer son propre menu. Nous verrons cette approche au chapitre présentant la création d'une extension du début à la fin.

Le mécanisme de sécurité nonce

Concept

WordPress possède différents mécanismes pour sécuriser les traitements de données dans le site. Les nonces sont une sécurité dite "de provenance".

Cela veut dire que WordPress va vérifier différents points :

- **L'authentification.** Est-ce que l'utilisateur est bien connecté ?
- **L'intention.** Faut-il protéger l'utilisateur de lui-même ?
- **La provenance.** D'où vient l'utilisateur ?

Ainsi, WordPress contrôle la validité des données *via* une clé unique, aléatoire et temporaire. C'est l'origine du mot "nonce" : *number used only once*, soit "un nombre utilisé une seule fois".

Prenons comme exemple l'effacement d'un article. L'utilisateur doit cliquer sur le lien Mettre à la Corbeille ; ce lien va contenir une clé unique et aléatoire pour cette action.

Lors du clic, le navigateur va envoyer une requête au serveur de votre site ; ce dernier va vérifier la présence de cette clé avant de supprimer l'article. Si la clé n'est pas trouvée, il retournera un message d'erreur.

La clé générée étant unique et limitée dans le temps, si vous copiez-collez le lien d'effacement d'un article et essayez ce lien 24 heures plus tard, l'effacement ne fonctionnera pas.

Le mécanisme permet ainsi de lutter contre les attaques de hackers, mais également contre les mauvaises manipulations du logiciel. Ce mécanisme a été introduit à la version 2.0.3 de WordPress. Depuis il sécurise toutes les fonctionnalités du logiciel.

Enfin, les nonces fonctionnent aussi bien avec les formulaires de données que les liens HTML (les données HTTP `POST` et `GET`).

Fonctions

L'API nonce comprend cinq fonctions.

Vous retrouvez deux fonctions génériques `wp_create_nonce()` et `wp_verify_nonce()`, deux fonctions dédiées à la génération du nonce, `wp_nonce_field()` et `wp_nonce_url()`, sécurisant respectivement les données `POST` et `GET`, et enfin une dernière fonction `check_admin_referer()` gérant le contrôle de validité du nonce.

wp_create_nonce($action = -1)

Cette fonction permet de générer la valeur unique du nonce. Elle prend comme seul paramètre le nom de l'action du nonce, ce qui donne un contexte à la clé générée.

Par exemple, pour l'effacement d'un article, le nom du nonce pourrait être `delete_post`. Cette fonction ne fait que retourner la valeur du nonce.

wp_verify_nonce($nonce, $action = -1)

Cette fonction permet de vérifier la validité d'un nonce. Elle prend deux paramètres : le premier est la valeur du nonce, le second est le nom du nonce que l'on souhaite vérifier (ou son contexte).

Si nous voulons valider le nonce lors de l'effacement d'un article, nous passerons comme second paramètre `delete_post`, soit la même action que la création.

La valeur du nonce est récupérée manuellement *via* une variable `$_GET` ou `$_POST`.

wp_nonce_field($action = -1, $name = "_wpnonce", $referer = true, $echo = true)

Contrairement aux deux premières fonctions, `wp_nonce_field()` est dédiée à un usage spécifique : générer un nonce dans un formulaire de données `POST`. La fonction va automatiser la création d'un champ `input` de type `hidden`, il suffit juste de préciser le nom de l'action.

Cette fonction possède cependant quatre paramètres :

- Le nom de l'action à sécuriser.
- Le nom utilisé par le champ HTML `input`.
- L'ajout ou non d'un champ `input` de type `hidden` contenant le référent de la page.
- L'écriture ou non du code HTML. On a le choix d'afficher le code HTML avec la fonction PHP `echo()` ou bien de retourner la valeur.

Généralement, les paramètres par défaut suffisent, et on ne complète que le premier paramètre.

wp_nonce_url($actionurl, $action = -1)

Cette fonction permet de sécuriser les données envoyées avec une requête HTTP `GET`. Autrement dit, cela permet de sécuriser les liens dans WordPress.

Elle ne prend que deux paramètres : le premier est l'adresse HTTP à sécuriser, le second paramètre est le nom de l'action qu'on va attribuer au nonce.

Cette fonction retourne toujours comme valeur l'adresse sécurisée.

check_admin_referer($action = -1, $query_arg = '_wpnonce')

Pour vérifier la validité d'un nonce, nous avons vu qu'il existe la fonction `wp_verify_nonce()`. La fonction `check_admin_referer()` est là pour simplifier les contrôles et normaliser les messages d'erreur.

Elle prend comme paramètres le nom de l'action et optionnellement le nom de la valeur à récupérer dans les données `POST` ou `GET`. Ce second paramètre n'est modifié que dans de très rares cas.

Cette fonction se place généralement avant l'affichage de WordPress, souvent lors de l'action `init` ou `admin_init`. Elle contrôle la présence et la validité du nonce ; le cas échéant, elle coupe le script PHP avec la fonction `wp_die()` et affiche un message d'erreur.

Exemples

Ajout du nonce dans un formulaire HTML :

```
<form method="POST" action="">
```

```
...
  <p class="submit">
    <?php wp_nonce_field('options_monplugin'); ?>
    <input type="submit" name="update_options_monplugin" value="Enregistrer les
    modifications" />
  </p>
</form>
```

Ajout du nonce dans une adresse Internet :

```
<?php
$lien = 'adresse-du-plugin.php';
$lien = wp_nonce_url($lien, 'lienaction_monplugin');
?>
<a href="<?php echo $lien; ?>">Lien sécurisé</a>
```

Contrôle du nonce avec `check_admin_url()`, qui correspond au premier exemple avec le formulaire `POST` :

```
// On contrôle les données avant l'affichage de WordPress
add_action('admin_init', 'check_data_plugin');

function check_data_plugin() {
  // On vérifie que les données du formulaire ont été envoyées
  if ( isset($_POST['update_options_monplugin']) ) { // Si oui
    // Contrôle de sécurité
    check_admin_referer('options_monplugin');

    // Suite de l'extension
    ...
  }
}
```

Bonnes pratiques

Pour ajouter la compatibilité avec les versions antérieures à WordPress 2.0.3, il faut utiliser la fonction PHP `function_exists()`.

L'utilisation des nonces dans les extensions est indispensable de nos jours, d'autant plus dans un contexte de développement pour professionnels. La sécurité est l'une des principales problématiques de notre temps…

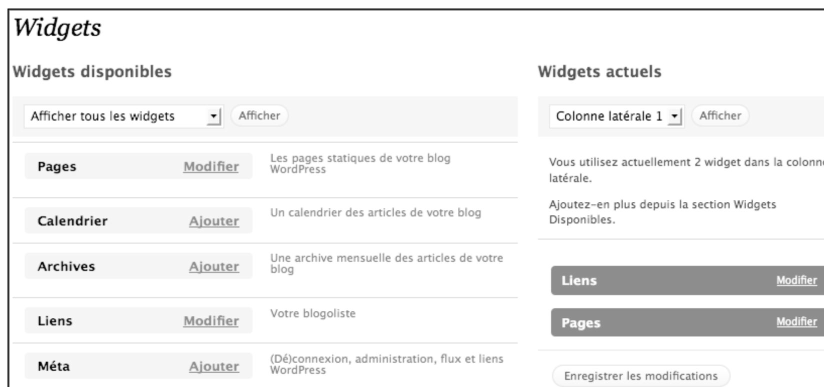
L'API des widgets

Concept

Les widgets ont fait leur apparition dans WordPress 2.2. Avant cette version, ils existaient sous la forme d'une extension. Lorsque l'on travaille avec les widgets de WordPress, le terme sidebar revient de façon récurrente. Par exemple, la colonne latérale placée à droite du thème par défaut est une sidebar.

Dans la logique des widgets, une sidebar n'est pas forcément une colonne latérale, c'est plus généralement un lieu spécifique dans le thème où l'on pourra ajouter et supprimer à la volée des blocs de contenus depuis l'interface d'administration (voir Figure 10.08). Ainsi, on peut retrouver les widgets également dans les colonnes en pied de page, ou dans le menu d'en-tête.

Figure 10.08
Les widgets vus depuis l'interface d'administration de WordPress.



Un widget dans WordPress comprend deux parties : un composant utilisateur et un composant administration. Le premier gère l'affichage de la fonctionnalité sur le thème, et le second permet de gérer un contrôle dans l'administration des widgets avec la configuration éventuelle d'options.

Fonctions

L'API de widgets propose de nombreuses fonctions, parmi lesquelles cinq sont indispensables pour la création des widgets.

register_sidebar([mixed args])

Cette première fonction permet d'enregistrer une sidebar dans le thème de WordPress. Elle prend comme seul paramètre un tableau PHP contenant différentes informations telles que :

- `name`, le nom de la sidebar dans l'interface d'administration ;
- `id`, le nom de l'ID HTML de la sidebar ;
- `description`, un descriptif de la sidebar ;
- `before_widget`, le code HTML qui précède chaque widget ;
- `after_widget`, le code HTML qui suit chaque widget ;
- `before_title`, le code HTML qui précède chaque titre de widget ;
- `after_title`, le code HTML qui suit chaque titre de widget.

D'un point de vue PHP, l'ordre d'utilisation des valeurs est le suivant :

```
echo $before_widget; // Début du widget
echo $before_title . 'Titre du widget' . $after_title; // Le titre entouré des
variables before et after
// Contenu du widget
echo $after_widget; // Fin du widget
```

Notez que les informations `name` et `id` peuvent contenir un marqueur `%d` ; ce dernier sera remplacé par le numéro de la sidebar. Ce numéro est automatiquement incrémenté de 1 pour chaque nouvelle sidebar.

Cette fonction est à utiliser dans le fichier functions.php du thème.

register_sidebars($number = 1, $args = array())

Cette deuxième fonction permet d'enregistrer plusieurs sidebars en une fois. Le premier paramètre est le nombre de sidebars, et le second paramètre correspond au tableau de données passé à la fonction `register_sidebar()`.

Il est important pour cette fonction de ne pas oublier l'utilisation des marqueurs `%d` dans le nom de la sidebar et l'ID, pour différencier correctement les sidebars. Par exemple :

```
register_sidebars(2, array('name'=>'Zone %d'));
```

... permet de créer deux sidebars, nommées Zone 1 et Zone 2.

De fait, cette fonction est à éviter dans le cas où vous voulez donner un nom unique à chaque sidebar, auquel cas plusieurs appels à `register_sidebar()` sont préférables.

Cette fonction est à utiliser dans le fichier functions.php du thème.

dynamic_sidebar($index = 1)

Cette fonction permet d'imprimer le contenu d'une sidebar. Elle est placée directement dans le thème, à l'endroit où on désire insérer les widgets.

Par exemple, dans le thème par défaut de WordPress, vous retrouvez cette fonction dans le fichier sidebar.php.

Elle prend comme seul paramètre le numéro ou le nom de la sidebar. Cette fonction est généralement utilisée sous forme de condition. Si la fonction existe et si la sidebar contient des widgets, alors on l'affiche. Sinon on affiche un code HTML contenu dans le thème.

Cela donne le code suivant :

```
<?php if ( !function_exists('dynamic_sidebar') || !dynamic_sidebar() ) : ?>
   ...
   Si la fonction n'existe pas ou qu'il n'y a pas de widget dans la sidebar, on
affiche un code HTML
  ...
<?php endif; ?>
```

wp_register_sidebar_widget($id, $name, $output_callback, $options = array())

Cette fonction permet d'enregistrer un widget dans WordPress. Elle est spécifique à la partie utilisateur du widget, et elle permet de spécifier la fonction à exécuter lors de l'affichage de la sidebar.

Elle prend quatre paramètres :

- Un ID unique pour chaque widget. C'est indispensable pour permettre les effets JavaScript de l'interface d'administration. Si l'ID existe déjà, seul le dernier enregistrement de widget sera conservé.
- Le nom du widget.
- La fonction utilisée par le widget pour l'affichage côté utilisateur.
- Un tableau PHP pour spécifier des options dont :
  - le `classname` permettant de changer le nom de la classe HTML lors de l'affichage du widget ;
  - la description affichée dans l'interface d'administration de WordPress à proximité du nom du widget.

wp_register_widget_control($id, $name, $control_callback, $options = array())

Cette dernière fonction va de pair avec `wp_register_sidebar_widget()` ; elle permet d'enregistrer dans WordPress le contrôle du widget dans l'interface d'administration.

Elle prend comme paramètres le même ID et le même nom utilisés dans la fonction précédente. Le troisième paramètre est le nom de la fonction à exécuter dans le contrôle du widget. Enfin elle peut prendre un tableau d'options permettant la spécification de la largeur du contrôle dans l'interface d'administration.

Le quatrième paramètre est très peu utilisé.

La classe *WP_Widget*

L'API de widgets a fortement évolué lors de la version 2.8, en cherchant à simplifier le développement de widgets par l'utilisation de la classe PHP `WP_Widget`. Désormais, pour créer un widget, il suffit d'étendre cette classe et ses fonctions.

```
class MonWidget extends WP_Widget {

  function MonWidget() {
    // les fonctionnalités spécifiques de votre widget
  }

  function form($instance) {
    // le code qui affichera le formulaire d'administration.
  }

  function update($new_instance, $old_instance) {
    // le code qui s'assure que les options sont enregistrées
```

```
        // lors d'une modification.
      }

      function widget($args, $instance) {
        // le code qui affichera la partie publique du widget
      }
    }
    register_widget('MonWidget');
```

Nous verrons un exemple pratique dans un prochain chapitre.

Bonnes pratiques

Les trois premières fonctions vues ci-dessus sont généralement utilisées dans les thèmes, tandis que les deux dernières servent dans les extensions.

Cependant, vous pouvez utiliser les fonctions dans les deux parties sans aucun problème, c'est juste une question d'élégance dans le code.

La déclaration des widgets se positionne logiquement dans la partie extensions et interface d'administration de WordPress, tandis que le thème déclare les fonctions spécifiques aux besoins de sa structure HTML.

Enfin, la méthode `register_sidebars()` est rarement utilisée car avec cette fonction vous ne pouvez nommer explicitement chaque sidebar.

Vous vous retrouvez obligatoirement avec des noms incrémentés comme colonne1, colonne2…

Il est impossible alors de spécifier colonne de gauche, colonne de droite, etc. C'est pour cela qu'on utilise et déclare généralement plusieurs fois la fonction `register_sidebar()`.

Ressources

L'API des widgets est documentée dans le Codex anglais de WordPress : **http://codex.wordpress.org/Widgets_API**.

Vous pouvez également trouver un modèle de widget accompagné d'un tutoriel complet à l'adresse suivante :

http://justintadlock.com/archives/2009/05/26/the-complete-guide-to-creating-widgets-in-wordpress-28.

Classe d'accès à la base de données – WPDB

Concept

WordPress est un logiciel open-source développé en PHP et fonctionnant avec la base de données libre MySQL. Cette base de données est le lieu de stockage de toutes les données de

votre site ; c'est pour cette raison qu'à chaque mise à jour de WordPress, il est recommandé de faire une sauvegarde de la base de données, afin de préserver le contenu de votre site.

Mis à part les médias que vous publiez sur votre site (photos, sons, vidéos), tout le reste est stocké en un lieu central, la base de données MySQL.

Pour travailler avec cette base de données, WordPress possède une classe de connexion MySQL, inspirée des travaux de Justin Vincent sur ezSQL. Cette classe est instanciée une seule fois dans WordPress. Il n'est pas nécessaire d'instancier une connexion par requête SQL. De ce fait, pour effectuer une requête SQL, il faut travailler avec la variable globale `$wpdb`. Cette variable est une instanciation de la classe WPDB.

Elle donne accès à différentes méthodes pour récupérer du contenu depuis la base SQL. Nous allons voir en détail ces fonctions.

Fonctions

Dans un premier temps, nous allons voir les cinq méthodes présentes historiquement dans WordPress. Puis nous verrons les nouvelles méthodes et ce qu'elles apportent.

Rappel SQL. On peut comparer une table de base de données SQL à un tableau (voir Figure 10.09) ; ce tableau contient des lignes, des colonnes et des cellules. Pour s'en convaincre, le plus simple est d'utiliser le logiciel phpMyAdmin et de naviguer dans le contenu des tables de la base.

Afficher : 30 enregistrement(s) à partir de l'enregistrement n° 0
en mode horizontal et répéter les en-têtes à chaque groupe de 100
Trier sur l'index: aucune

←T→			ID	post_author	post_date	post_date_gmt	post_content	post_title	post_category	post_excerpt
			1	1	2008-10-23 22:52:09	2008-10-23 20:52:09	Welcome to WordPress. This is your first post. Edi...	Hello world!	0	
			2	1	2008-10-23 22:52:09	2008-10-23 20:52:09	This is an example of a WordPress page, you could ...	About	0	

Figure 10.09
Aperçu de données stockées dans MySQL avec l'outil phpMyAdmin.

get_var($query=null, $x = 0, $y = 0)

Cette fonction permet de récupérer une information précise de la base de données, autrement dit la valeur d'une cellule spécifique.

En général, cette méthode est utilisée pour retourner une information spécifique ; d'ailleurs le type de retour est une chaîne de caractères, par exemple la valeur d'une option de WordPress.

Le premier paramètre est la requête SQL à effectuer. Si cette dernière retourne plus d'une ligne ou plus d'une colonne, il est possible de spécifier le numéro de lignes et de colonnes respectivement avec les deuxième et troisième paramètres.

get_row($query = null, $output = OBJECT, $y = 0)

Cette fonction permet de récupérer une ligne précise de la base de données, par exemple retrouver les données d'un article spécifique.

Cette méthode prend trois paramètres : le premier est la requête SQL à effectuer. Le deuxième est le type de sortie qu'on souhaite obtenir. Nous détaillons les différents types de retour dans la section suivante.

Si le résultat de la requête retourne plusieurs lignes, il est possible de choisir la ligne qu'on souhaite utiliser avec le troisième paramètre.

get_col($query = null, $x = 0)

Cette fonction permet de récupérer les données d'une colonne. Par exemple, lorsque nous souhaitons obtenir la liste complète des ID d'articles uniquement de notre base de données.

La méthode prend deux paramètres : le premier est la requête SQL à effectuer, le second est le numéro de la colonne à utiliser dans le cas où la requête SQL retourne plusieurs colonnes.

get_results($query = null, $output = OBJECT)

Cette fonction permet de récupérer un ensemble de résultats, autrement dit plusieurs lignes et plusieurs colonnes de données.

Par exemple, lorsqu'on souhaite récupérer les données des cinq derniers articles, on récupère les cinq dernières lignes qui correspondent aux cinq articles et toutes les colonnes qui correspondent à toutes les informations des articles.

Le premier paramètre est la requête SQL à effectuer, le second paramètre est le type de retour que l'on souhaite obtenir.

query($query)

Contrairement aux fonctions précédentes, la méthode `query()` ne permet pas de récupérer de résultats.

Elle est utilisée uniquement pour effectuer des modifications sur la base de données SQL, autrement dit un ordre SQL différent de `SELECT`.

Ainsi, on retrouve des requêtes SQL du type `UPDATE`, `INSERT`, `DELETE`, `ALTER`, etc.

escape($string|$array)

Cette méthode permet de sécuriser une chaîne de caractères ou un tableau PHP avant son insertion en base de données. En réalité, la méthode `escape` ne fait qu'utiliser la fonction PHP `addslashes()`.

Les développeurs voudraient remplacer cette fonction générique par une fonction spécifique à la sécurité contre les injections SQL, mais la fonction `mysql_real_escape_string()` de PHP pose de sérieux problèmes de compatibilité chez les hébergeurs.

Cette méthode dispose par ailleurs d'un alias depuis WordPress 2.8 : `esc_sql()`.

prepare($query=null [, $args, $args, ...])

Cette méthode permet d'automatiser la sécurisation des données avant la requête SQL. Cette fonction utilise la méthode `escape()` pour la sécurisation.

Pour comprendre l'intérêt de cette méthode, rien ne vaut un exemple. Avant `prepare()`, pour sécuriser une fonction, on faisait :

```
function jajoute( $valeur1,  $valeur2 ){
  global $wpdb;

  // On sécurise
  $valeur1 = $wpdb->escape($valeur1);
  $valeur2 = $wpdb->escape($valeur2);

  // On exécute la requête
  $wpdb->query("INSERT INTO ma_table ( 'champ1', 'champ2' ) VALUES (
'$valeur1', '$valeur2' )");
}
```

Avec la méthode `prepare()`, cela donne :

```
function jajoute( $valeur1,  $valeur2 ){
   global $wpdb;


  // On sécurise et on exécute la requête
  $wpdb->query( $wpdb->prepare("INSERT INTO ma_table ( 'champ1', 'champ2' )
VALUES ( %s, %s )", $valeur1, $valeur2 ) );
}
```

Les deux fonctions font la même chose, mais on voit tout de suite que la méthode `prepare()` permet un gain de temps lors du développement et améliore la lisibilité du code de votre extension.

insert($table, $data, $format = null)

Cette fonction automatise encore un peu plus l'insertion de données dans la base SQL.

Plutôt que d'écrire et sécuriser la requête SQL d'insertion, il est possible d'utiliser la fonction `insert()` qui se charge de toute cette partie.

Cette fonction prend deux paramètres : le premier est le nom de la table dans laquelle on souhaite ajouter les données, le second paramètre doit être le tableau associatif contenant les valeurs à ajouter.

Par exemple, nous disposons d'une table appelée `wp_monplugin`. Elle possède trois informations : `id`, `name` et `active` (ID, le nom et son état).

Vous utiliserez `insert()` de la façon suivante :

```
insert( 'wp_monplugin', array('id' => 123, 'name' => 'Le nom', 'active' =>
true) )
```

Ainsi, vous êtes capable d'insérer une ligne de contenu sans écrire un mot de SQL.

Le dernier argument précise le format des données, à choisir entre `'%d'` (un nombre décimal) et `'%s'` (une chaîne). Par défaut, les données sont traitées comme des chaînes.

update($table, $data, $where, $format = null, $where_format = null)

La fonction `update()` est très semblable à la fonction `insert()`, à la différence près qu'elle réalise un ordre SQL `UPDATE` et non `INSERT`. Cela sous-entend un paramètre en plus. En effet, contrairement à un ordre `INSERT`, un ordre `UPDATE` peut être filtré *via* la condition `WHERE`.

Vous retrouvez donc une fonction avec trois arguments : le nom de la table, les données à mettre à jour, et enfin les conditions dans le `WHERE`.

Petit exemple, nous souhaitons mettre à jour les données de la ligne ayant l'ID 3 de la table `wp_monplugin` :

```
update( 'wp_monplugin', array('name' => 'Nom modifié', 'active' => false),
array('id' => 3) )
```

Là encore, vous êtes capable d'effectuer un ordre `UPDATE` sans écrire la moindre ligne de SQL.

L'argument optionnel `$format` fonctionne de la même manière que pour la méthode `insert`. De son côté, `$where_format` est un équivalent de `$format`, mais cette fois pour les valeurs de `$where` plutôt que de `$data`.

replace($table, $data, $format = null)

Nouveauté de WordPress 3.0, la méthode `replace` ne surprendra pas les habitués de MySQL : elle insère une ligne dans la table visée (comme `insert`) mais, si une ligne existante de la table a la même valeur de clé primaire ou d'index, la ligne existante est effacée avant l'insertion de la nouvelle ligne.

```
replace( 'wp_monplugin', array( 'column' => 'lapin', 'field' => 'malin' ) )
```

tables($scope = all, $prefix = false, $blog_id = 0)

Autre nouveauté de la version 3.0, cette méthode permet de récupérer le contenu des tables de WordPress (ou d'une partie de celles-ci) dans un tableau PHP, afin de les traiter à votre manière. Par défaut, la méthode sans argument renvoie un tableau avec l'intégralité des tables de WordPress.

L'argument `$scope` permet de cibler plus précisément des tables :

- `'blog'` renvoie les tables liées au site courant (en cas d'utilisation multisite).
- `'global'` renvoie les tables de tous les sites du réseau (en cas d'utilisation multisite).
- `'old'` renvoie les tables obsolètes.
- `'all'` renvoie l'ensemble des tables actuelles : site courant, reste du réseau (par défaut).

`$prefix` précise si l'on souhaite récupérer le préfixe du site.

Enfin, `$blog_id` précise l'ID du site à préfixer. Par défaut, il s'agit du site principal.

Les différents types de retour

La classe WPDB possède trois constantes prédéfinies pour choisir le type de retour des résultats SQL.

OBJECT (par défaut)

Les résultats se présenteront sous la forme d'un tableau d'objets.

Exemple :

```
$result->name
```

ARRAY_A

Les résultats se présenteront sous la forme d'un tableau associatif.

Exemple :

```
$result['name']
```

ARRAY_N

Les résultats s'afficheront sous la forme d'un tableau numéroté classique.

```
$result[2]
```

2 étant la position du champ `name` dans le tableau.

Généralement, le type `object` convient à tous les usages. Cependant, dans des cas particuliers, il peut être pratique d'obtenir les résultats dans un tableau associatif (pour fusionner des tableaux, les ordonner, etc.).

Enfin, la troisième constante `ARRAY_N` est très peu employée ; il n'y a d'ailleurs aucune fonction dans WordPress qui l'utilise.

Créer une table dans WordPress

Avec des extensions assez complexes, il peut arriver de devoir créer une table dans la base de données pour stocker des informations spécifiques à l'extension. L'arrivée des taxinomies personnalisées et des types de contenu personnalisé dans la version 3.0 limite désormais le besoin de recourir à de nouvelles tables à chaque besoin spécifique, mais la possibilité reste non négligeable.

La création de table dans WordPress a généralement lieu lors de l'activation de l'extension. La méthodologie classique consiste à tester l'existence de la table, et si elle n'existe pas, la requête de création est effectuée.

Il existe une autre façon de procéder, cette fois-ci avec les options. L'extension stocke en base de données le numéro de version de la base de données. À chaque modification de la base de données, lors des évolutions de l'extension, le développeur incrémente le numéro de version. L'extension se charge ensuite de comparer chaque fois les deux numéros de version et éventuellement de lancer la fonction de création.

Pour y parvenir, nous allons utiliser la fonction `dbDelta()`. Cette dernière permet de sécuriser et de vérifier la conformité de la requête de création de table.

Exemple :

```
// Fonction à lancer lors de l'activation de l'extension
function create_table_wp () {
  global $wpdb;

  // On construit le nom de la table avec le préfixe de WordPress
  $table_name = $wpdb->prefix . "monplugin";

  // On teste la présence de la table
  if($wpdb->get_var("show tables like '$table_name'") != $table_name) {

    // On construit la requête SQL avec le nom de la table
    $sql = "
      CREATE TABLE " . $table_name . " (
        id INT(9) NOT NULL AUTO_INCREMENT,
        name VARCHAR(255) NOT NULL,
        active INT(1) NOT NULL,
        UNIQUE KEY id (id)
      );
    ";

    // On inclut une librairie de WordPress contenant la fonction dbDelta.
    La librairie n'est pas lancée depuis la partie client, raison pour laquelle
    on prend cette précaution.
    require_once(ABSPATH . 'wp-admin/includes/upgrade.php');

    // On exécute la requête SQL.
    dbDelta($sql);

  }
}
```

Faites attention de ne pas entourer le nom de la table de *simple quote* (apostrophe), cela fausserait la requête SQL.



Attention

À moins que vous ne connaissiez bien le langage SQL, écrire le code SQL de création de table est un exercice assez difficile. À réserver donc aux initiés… et évitez de recopier du code sans savoir exactement ce qu'il fait.

Pour vous aider, nous vous suggérons de créer votre table depuis l'outil phpMyAdmin (voir Figure 10.10). Vous pourrez ainsi facilement construire votre table en choisissant visuellement les types, noms, valeurs par défaut, clés primaires, index et autres options d'incrémentation.

Figure 10.10
Création d'une table MySQL depuis l'outil phpMyAdmin.

localhost ▸ wp27 ▸ wp_matable

Champ	Type	Taille/Valeurs*1	Interclassement	Attributs	Null	De
id	VARCHAR	10	big5_chinese_ci		not null	
blog_id	INT	255		BINARY	not null	
name	VARCHAR	20			not null	
	VARCHAR				not null	
	VARCHAR				not null	

Commentaires sur la table: | Moteur de stockage: InnoDB | Interclassement: ascii_general_ci

Sauvegarder | Ou Ajouter 1 champ(s) | Exécuter

Une fois que votre table est prête, il suffit d'exporter sa structure, toujours depuis phpMyAdmin, pour obtenir le code SQL.

Bonnes pratiques

Nous avons vu qu'il existe différentes méthodes pour accéder aux données ; l'objectif de cette diversité est de faciliter les appels SQL dans le code.

Alors bien sûr, il est possible d'utiliser la méthode `get_results()` pour tous les types de `SELECT`, mais un bon développeur cherchera toujours à employer la bonne méthode selon le résultat attendu de la requête SQL. Cela évitera l'utilisation hasardeuse des paramètres X et Y.

Nous pouvons également nous interroger sur l'utilité des méthodes `prepare()` et `escape()`, étant donné que les fonctions `insert()` et `update()` sont plus simples d'utilisation…

En réalité, ces méthodes sont utiles dans de nombreux cas. Une requête SQL n'est pas toujours contenue dans une chaîne de caractères, parfois elle peut être construite à la volée *via* différentes conditions.

C'est par exemple le cas de la requête de la classe `WP_Query` de WordPress. Étant donné que la requête SQL des articles diffère selon la page où l'on se trouve, il est nécessaire de sécuriser les données différemment, avec la méthode `escape()` par exemple !

Le cache de WordPress

Concept

WordPress possède une classe de cache. Cette dernière ne propose pas un cache comme on l'entend le plus souvent. Il ne s'agit pas de créer une copie HTML d'une page de WordPress et de l'afficher aux visiteurs – ceci étant le mode de fonctionnement d'extensions comme WP-Cache et WP Super Cache (**http://wordpress.org/extend/plugins/wp-super-cache/**).

Le cache de WordPress est un cache objet. Au lieu de mettre en cache le résultat final du traitement PHP en stockant une page HTML, le cache objet de WordPress va stocker les résultats provenant de la base de données.

En effet, les problématiques de montée en charge ont toujours comme cause une utilisation excessive des requêtes SQL. Il n'est pas rare de trouver des sites générant plus de 100 requêtes pour afficher une simple page PHP.

Ces requêtes proviennent généralement d'une extension ou d'un thème mal développés. Il se peut aussi qu'avec un nombre raisonnable de requêtes (20 à 30) un site avec une très forte fréquentation ait du mal à tenir la charge.

Pour améliorer les performances, les développeurs souhaitent donc limiter les accès en base de données *via* un cache. L'avantage de ce type de cache, contrairement à un cache complet HTML, c'est qu'on conserve les fonctionnalités avancées de type notation, compteur de vues, sondage, gestion de membres…

Les développeurs auraient pu imposer un mécanisme de stockage du cache, dans des fichiers par exemple, mais il n'en est rien. La classe de cache peut être très facilement remplacée par une solution alternative.

Cela permet de choisir le type de cache voulu. Il existe ainsi des classes de cache pour travailler avec :

- l'extension PHP APC ;
- l'extension PHP XCache ;
- l'extension PHP eAccelerator ;
- l'extension PHP MemCache avec le logiciel MemCached.

Cette liste n'est pas exhaustive, il vous est possible de stocker les données où bon vous semble. Rien n'empêche de créer une classe de cache pour stocker le cache dans des fichiers situés sur un serveur optimisé pour ce type d'accès.

Par défaut, WordPress stocke les données directement dans la mémoire de PHP, cela évite de faire plusieurs fois la même requête SQL dans le traitement d'une page PHP. C'est très pratique pour les performances des options. Ces dernières sont en effet utilisées un peu n'importe comment par un grand nombre d'extensions et de thèmes.

Cependant, cette implémentation se veut minimaliste, car les données stockées dans la mémoire de PHP ne sont conservées que durant l'exécution de la page. Autrement dit, par défaut, le cache de WordPress ne sert pas à grand-chose car il n'est pas commun à tous les utilisateurs.

D'où l'utilité de le remplacer par une des solutions alternatives présentées ci-dessus.

Activation du cache

Depuis les dernières versions de WordPress, le cache est activé automatiquement.

Sur les anciennes versions, pour activer le cache, il fallait éditer le fichier de configuration de WordPress, wp-config.php, et y ajouter la ligne :

```
define('ENABLE_CACHE', true);
```

Attention, la constante `WP_CACHE`, rajoutée par les extensions WP-Cache, WP Super Cache, ne doit pas être définie.

Remplacer l'implémentation du cache de WordPress par une alternative

Comme nous l'avons expliqué ci-dessus, l'implémentation initiale du cache de WordPress est peu intéressante.

Pour remplacer la classe de cache de WordPress par la vôtre, il suffit de placer le fichier PHP de votre classe de cache dans le dossier wp-content et de le nommer object-cache.php.

Fonctions

La plupart des fonctions de la classe de cache prennent les arguments suivants :

- `$key` ou `$id` : la clé unique d'une valeur dans le cache.
- `$data` : la valeur des données à stocker.
- `$flag` : ce paramètre est optionnel, il permet de définir un groupe pour les données que vous stockez. Cela permet de structurer le cache. Par exemple, si vous enregistrez des données relatives aux articles, vous utiliserez le groupe posts.
- `$expire` : la durée de vie en secondes du cache (par défaut 900).

Ci-après, vous trouverez les fonctions disponibles *via* l'API de cache de WordPress.

wp_cache_add($key, $data, $flag = '', $expire = 0)

Cette première fonction permet d'ajouter des données dans le cache. Elle teste cependant la présence de la clé dans le cache. Si la clé n'existe pas, alors la valeur est enregistrée dans le cache, sinon la fonction ne fait rien et retourne la valeur `false`. Il est possible de spécifier le groupe et la durée avec expiration.

wp_cache_delete($id, $flag = '')

Cette fonction permet d'effacer une entrée du cache. Elle prend en paramètres l'identifiant à effacer et son groupe.

wp_cache_get($id, $flag = '')

Cette fonction retourne la valeur de l'entrée du cache. Elle prend comme paramètres l'identifiant et son groupe. Si l'entrée n'existe pas, ou si elle a expiré, la fonction renvoie `false`.

wp_cache_replace($key, $data, $flag = '', $expire = 0)

Cette fonction permet de remplacer la valeur d'une entrée existante du cache par une nouvelle valeur. Si la valeur n'existe pas, la fonction retourne `false`.

wp_cache_set($key, $data, $flag = '', $expire = 0)

Cette fonction permet de définir la valeur d'une entrée du cache. Si la valeur existe déjà, elle est écrasée par la nouvelle, et si l'entrée n'existe pas, elle est créée.

wp_cache_init()

Cette fonction permet d'initialiser un nouvel objet pour la classe de cache. Elle est automatiquement appelée par WordPress si le cache est activé dans le fichier de configuration.

wp_cache_flush()

Cette fonction permet d'effacer le contenu du cache.

wp_cache_close()

Cette fonction ne réalise rien. Elle est présente à des fins de rétrocompatibilité avec les anciennes extensions.

Utiliser le cache dans son extension

Il est possible de mettre en cache le résultat d'une requête SQL, mais il est également possible de mettre en cache une page HTML que l'on a récupérée d'un site distant.

Par exemple, vous allez récupérer le code HTML de la page **http://www.herewithme.fr** :

```
// On essaie de récupérer la valeur du cache.
$actus = wp_cache_get('actus', 'groupe-ext');

// Si la valeur n'existe pas, on entre dans la condition
if( $actus == false ) {
  // On récupère la page HTML avec la classe Snoopy.
  // Précision, l'objectif n'est pas de savoir ce que fait Snoopy…
  $snoopy = new Snoopy;
  $snoopy->fetch('http://www.herewithme.fr/');
  $actus = $snoopy->results;

  // Stocke le résultat en cache
  wp_cache_set('actus', $actus, 'groupe-ext');
}
```

Le processus se déroule en deux étapes :

- Vous essayez de récupérer la valeur depuis le cache.
- Si elle n'existe pas, vous effectuez le traitement gourmand en performances et vous stockez le résultat dans le cache.

Bonnes pratiques

Il ne faut pas utiliser le cache de WordPress à toutes les sauces. Par exemple, cela ne sert à rien de mettre en cache le résultat de fonctions propres à WordPress. Ces dernières sont généralement déjà développées de façon à utiliser correctement le cache de WordPress.

Le cache doit principalement être employé dans deux cas :

- le stockage des résultats SQL d'une requête écrite dans une extension ;
- le stockage de données provenant d'un site tiers.

Ressources

Nous allons faire un petit tour des principales classes alternatives de cache, avec les points forts de chaque technologie.

XCache

http://dougal.gunters.org/blog/2008/08/29/xcache-object-cache-plugin-for-wordpress-25

XCache est une extension pour PHP. C'est un cache OPcode optimisant les traitements PHP. Il fait l'objet d'un développement soutenu ; c'est l'un des projets les plus récents dans ce domaine.

Memcached

http://wordpress.org/extend/plugins/memcached/

MemCache est une extension PHP. Elle permet de travailler avec le logiciel MemCached. C'est un logiciel permettant de gérer un cache, mais de façon beaucoup plus générique qu'une extension PHP. MemCache n'optimise pas par exemple le code PHP. L'intérêt de MemCache, comparé au cache OPcode, c'est qu'il peut être réparti sur différents serveurs *via* du load balancing. Cela permet une meilleure répartition de la charge.

APC

http://www.lazybrain.de/wordpress-apc-object-cache-modifiziert.html

APC ressemble à XCache. C'est une extension pour PHP qui joue le rôle de cache et d'optimisation OPcode. APC sera vraisemblablement intégré en natif dans la prochaine version de PHP. Il a l'avantage de pouvoir filtrer les fichiers passant par l'optimisation OPcode.

Cela permet de filtrer le fichier kses.php de WordPress, ce dernier générant régulièrement des erreurs fatales aux processus PHP sur les sites à haute fréquentation.

La taxinomie dans WordPress

C'est quoi la taxinomie ?

La taxinomie, taxonomy en anglais, désigne une méthode de classification des informations dans une architecture structurée de manière évolutive. Le terme est couramment employé pour des systèmes de gestion de contenu (CMS), dont WordPress fait partie.

Source : **Wikipedia.fr**.



Il y a deux écritures pour ce terme en français : taxinomie et taxonomie. Nous utiliserons de préférence le mot taxinomie dans ce livre, cependant rien ne vous empêche d'employer l'autre écriture.
Pour plus d'informations, allez voir à cette adresse :
http://blogokat.canalblog.com/archives/2005/11/07/968967.html.

Implémentation dans WordPress

Jusqu'à la version 2.3, WordPress disposait d'une vision limitée de la taxinomie. Pour s'en convaincre, il suffit de visionner la base de données de WordPress de l'époque. Cette dernière gérait distinctement les catégories des articles de celles des liens, et les tags n'étaient pas présents par défaut dans WordPress.

Tout cela a bien changé… Depuis WordPress 2.3, la base de données a évolué vers un schéma de taxinomie plus évolué et surtout complètement générique.

Les tables de catégories des articles, des liens et des liaisons, ont été supprimées au profit de trois tables complètement génériques.

Ces trois tables sont les suivantes :

- `wp_terms` ;
- `wp_term_relationships` ;
- `wp_term_taxonomy`.

Elles seront détaillées dans la section suivante.

Une dernière chose concernant la taxinomie dans WordPress : par défaut il en existe trois ; ce sont les catégories d'articles, les catégories de liens et les tags d'articles.

WordPress met à disposition une API pour étendre les taxinomies qui permet l'ajout, la modification, la suppression, etc.

La table *wp_terms*

Cette table contient la totalité des termes de WordPress. Lorsque vous ajoutez une catégorie dans WordPress, en réalité vous ajoutez un terme dans la base de données et vous précisez le contexte.

Par exemple, si vous ajoutez une catégorie WordPress et un tag WordPress, vous obtiendrez dans la base de données un seul terme WordPress, bien qu'il soit utilisé dans deux contextes différents.

Cette table contient comme informations :

- l'ID du terme ;
- le nom du terme ;
- le slug ou identifiant du terme, utilisé lors de la construction des URL ;
- le groupe du terme (optionnel).

Cette table est en relation avec la table `wp_term_taxonomy`.

La table *wp_term_taxonomy*

Comme nous le précisions ci-dessus, un terme n'est ni une catégorie ni un tag. Un terme, c'est générique. C'est la table `wp_term_taxonomy` qui va spécifier le contexte du terme.

Le contexte, c'est par exemple un tag, une catégorie d'articles ou une catégorie de liens !

Cette table contient différentes informations :

- L'ID du contexte entre le terme et la taxinomie.
- L'ID du terme que l'on associe.
- L'identifiant de la taxinomie.
- Une description. Utilisée par exemple par les catégories dans WordPress.
- L'ID d'un contexte parent. Utilisé également par les catégories, pour gérer la hiérarchie.
- Le compteur d'utilisation. Employé par les tags et les catégories. C'est le nombre d'objets associés au contexte d'utilisation du terme. En d'autres termes, pour les catégories d'articles de WordPress, il s'agit du nombre d'articles classés dans la catégorie.

Cette table est en relation avec les tables `wp_terms` et `wp_term_relationships`.

La table *wp_term_relationships*

Cette dernière table permet de faire le lien entre les objets (lien, page, article) et le contexte d'utilisation du terme.

Elle comprend les informations suivantes :

- l'ID de l'objet ;
- l'ID du contexte d'utilisation.

Cette table est en relation avec la table `wp_term_taxonomy`.

Exemple

Pour bien comprendre, un exemple est nécessaire… Voilà la situation. Nous allons attribuer l'article ayant l'ID 9 dans la catégorie WordPress. Le terme WordPress est déjà utilisé dans la taxinomie tags de WordPress et possède l'ID 86.

Pour réaliser cette mise en relation, deux insertions en base de données sont nécessaires. La première étape va consister à créer le contexte de taxinomie Catégorie pour le terme WordPress.

Pour cela, il faut insérer une entrée dans la table `wp_term_taxonomy`, précisant la taxinomie `category`, vu qu'il s'agit d'une catégorie d'articles, et indiquer l'ID du terme (86).

Cette insertion dans la table de données va attribuer un ID unique à ce contexte. Ici l'ID sera 12.

La seconde insertion aura lieu dans la table de relation `wp_term_relationships`. Nous allons joindre le contexte d'utilisation à l'objet, dans ce cas l'article. L'insertion aura donc comme valeur 9 pour l'objet et 12 pour le contexte. Et voilà, la mise en relation est terminée.

Rassurez-vous… Toutes ces étapes sont déjà préprogrammées dans l'API de taxinomie mise à disposition dans WordPress !

Fonctions

Notez que le fichier wp-includes/taxonomy.php contenant l'API de taxinomie de WordPress est complètement documenté dans le code. N'hésitez pas à le consulter pour connaître l'intégralité des options et fonctions disponibles.

get_taxonomies($args = array(), $output = 'names', $operator = 'and')

Nouveauté de la version 3.0, cette fonction renvoie une liste des taxinomies enregistrées, sous forme d'un tableau PHP. Ce tableau peut contenir simplement les noms des taxinomies, mais, si le deuxième argument est autre chose que la chaîne `'names'`, le tableau contiendra des représentations de ces taxinomies sous forme d'objets PHP.

```
$taxonomies = get_taxonomies( '','names' );
foreach ( $taxonomies as $taxonomy ) {
  echo '<p>'. $taxonomy. '</p>';

}
```

get_object_taxonomies($object, $output = 'names')

Cette fonction permet de récupérer la liste des taxinomies utilisées par un type d'objet (article, lien, etc.).

Par défaut, dans WordPress, si vous passez 'post' comme paramètre à cette fonction, vous obtiendrez un tableau PHP contenant les taxinomies `category` et `post_tag`.

Vous pouvez récupérer la liste sous forme d'objets PHP plutôt que de nom en précisant `'objects'` pour le second paramètre.*taxonomy_exists($taxonomy)*

Cette fonction permet de tester l'existence d'une taxinomie. Elle prend comme seul paramètre le nom de la taxinomie à tester.

Si la taxinomie existe, la fonction retourne `true`, sinon c'est `false`.

Depuis la version 3.0, cette fonction remplace `is_taxonomy()`.

register_taxonomy($taxonomy, $object_type, $args = array())

Cette fonction permet d'ajouter une nouvelle taxinomie dans WordPress. Elle ne doit pas être utilisée avant l'événement `init`.

Elle prend trois paramètres :

- Le nom de la taxinomie.
- Le type d'objet que cible la taxinomie. Vous pouvez passer soit une chaîne de caractères, soit un tableau contenant plusieurs types d'objets.
- Un tableau d'options permettant de préciser différentes informations sur la taxinomie :
  - la hiérarchie ou non dans la taxinomie ;
  - la fonction de callback pour le compteur ;

– la réécriture ou non des URL dans WordPress ;

– l'ajout ou non du mot-clef dans la classe `WP_Query` de WordPress.

register_taxonomy_for_object_type($tax, $object_type)

Cette fonction ajoute une taxinomie existante à un type de contenu, classique ou personnalisé. Ainsi, après avoir créé une taxinomie particulière pour les articles, on pourra l'appliquer directement à un autre type avec cette fonction.

is_object_in_taxnonomy($objet_type, $taxonomy)

Détermine si un type de contenu est associé à une taxinomie donnée.

get_taxonomy_labels($taxonomy)

Cette fonction renvoie un objet PHP contenant tous les textes (ou labels) de la taxinomie nommée. Le tableau est associatif : il lie le nom d'un label (`name`, `singular_name`, `parent_item_colon`, `choose_from_most_used`, etc.) avec la chaîne correspondante de cette taxinomie.

get_objects_in_term($term_ids, $taxonomies, $args = array())

Cette fonction permet de récupérer un tableau d'objets, uniquement les ID, depuis trois critères :

- Un tableau PHP d'ID ou un ID de terme.
- Un tableau PHP de taxinomies ou une taxinomie.
- Un tableau PHP d'options. La seule présente ici est l'ordre des résultats : ascendant ou descendant.

get_term($term, $taxonomy, $output = OBJECT, $filter = 'raw')

Cette fonction permet de récupérer les données d'un terme. Elle prend quatre paramètres :

- L'ID du terme.
- La taxinomie.
- Le type de retour. Ce sont les mêmes que pour la classe WPDB.
- Permet de définir les filtres à activer selon l'utilisation (db, raw, display).

get_term_by($field, $value, $taxonomy, $output = OBJECT, $filter = 'raw')

Cette fonction permet de récupérer les données d'un terme, mais comparée à la fonction `get_term()`, il est possible de récupérer un terme depuis son slug ou le champ souhaité.

Ainsi, au lieu d'avoir un paramètre ID du terme, vous obtenez deux autres paramètres que sont :

- le champ utilisé pour récupérer le terme, le slug (identifiant) par exemple ;
- la valeur du champ.

get_terms($taxonomies, $args = '')

Cette fonction permet de récupérer un tableau de termes. Elle est principalement utilisée pour générer les listes de catégories et les nuages de tags.

Il est possible de spécifier la ou les taxinomies dans lesquelles vous souhaitez récupérer des termes. Vous pouvez également passer un tableau PHP en deuxième paramètre pour spécifier un grand nombre d'options, par exemple l'ordre, le sens des résultats.

Vous pouvez exclure, inclure certains termes, etc. Consultez le code source de la fonction pour voir les possibilités des options.

term_exists($term, $taxonomy = '', $parent = 0)

Cette fonction permet de tester la présence d'un terme dans WordPress. Pour cela, il faut préciser le nom ou l'ID du terme, ainsi que la taxinomie dans laquelle vous voulez tester sa présence. Vous pouvez également préciser le nom ou l'ID du terme parent, afin de limiter la recherche.

Si le terme existe, la fonction retourne son index de définition, sinon elle retourne la valeur 0 ou `false`.

Remplace `is_term()` depuis WordPress 3.0.

wp_delete_object_term_relationships($object_id, $taxonomies)

Cette fonction permet d'effacer les relations entre un objet et les contextes de taxinomie.

Elle prend deux paramètres : l'ID de l'objet et la taxinomie associée.

wp_delete_term($term, $taxonomy, $args = array())

Cette fonction permet d'effacer un terme dans WordPress. Elle est intelligente car si le terme n'est utilisé que dans une seule taxinomie et qu'on le supprime, WordPress efface le contexte, les relations et le terme.

Alors que si le terme est utilisé dans une autre taxinomie, WordPress n'efface que le contexte et les relations. Le terme est conservé pour l'autre taxinomie.

La fonction prend en arguments :

- L'ID du terme.
- La taxinomie.
- Un tableau PHP d'options. Aucune option n'est disponible pour la version 2.7 de WordPress.

wp_get_object_terms($object_ids, $taxonomies, $args = array())

Cette fonction permet de récupérer un tableau PHP de termes depuis l'ID d'un objet ou depuis un tableau d'objets.

Elle peut prendre trois arguments :

- l'ID d'un objet ou un tableau PHP ;

- la taxinomie que l'on cible ;
- un tableau d'options permettant de modifier l'ordre et le sens des résultats, ainsi que les champs qu'on souhaite récupérer (les ID, les noms ou toutes les données).

wp_set_object_terms($object_id, $terms, $taxonomy, $append = false)

Cette dernière fonction permet d'attribuer des termes à un objet dans une taxinomie définie, par exemple classer des catégories à un article.

Elle prend quatre paramètres :

- L'ID de l'objet.
- Le tableau des ID des termes à ajouter.
- La taxinomie sur laquelle on travaille.
- Ce dernier paramètre permet de définir si on ajoute les termes à ceux qui existent, ou si on remplace les termes existants.

L'URL rewriting de WordPress – *WP_Rewrite*

C'est quoi le rewriting ?

L'*URL rewriting* ou réécriture d'adresse Internet est une technique utilisée pour optimiser le référencement des sites dynamiques, autrement dit des gestionnaires de contenu (CMS) dont WordPress fait partie.

Les CMS ont longtemps été caractérisés par des URL complexes, comprenant en général un point d'interrogation, éventuellement le caractère &, ainsi que des noms de variables et des valeurs.

Exemple : **http://monsite.fr/article.php?id=12&category=5&tag=3**.

Le problème ici est que la plupart des moteurs de recherche n'indexent pas ce type d'adresse. Google, par exemple, n'indexe en général que les pages ayant au maximum deux paramètres dans l'URL : il n'indexe donc pas une page comme celle de notre exemple.

La technique d'URL rewriting va consister à réécrire ces adresses complexes en des adresses plus lisibles pour les humains et pleinement compatibles avec les moteurs de recherche.

Le résultat attendu sera de cet acabit : **http://monsite.fr/article-sur-la-politique**.

Le processus de réécriture est généralement effectué au niveau du serveur web *via* les fichiers .htaccess.

Implémentation dans WordPress

Ces fichiers .htaccess permettent de travailler sur les fonctionnalités du serveur web Apache 2. L'une de ces fonctionnalités est la réécriture des URL. Pour cela Apache 2 dispose d'un module : `mod_rewrite`.

Une petite recherche dans Google avec les termes `mod_rewrite` vous donnera toutes les ressources nécessaires à ce sujet.

Dans la plupart des CMS, la réécriture des URL se fait de façon complètement transparente dans le fichier .htaccess *via* des structures conditionnelles et un découpage des adresses.

Cela donne ce genre de résultat :

```
RewriteEngine on
RewriteRule ^index.html$ index.php [L]
RewriteRule ^forum-([0-9]+)-([0-9]+).html$ viewforum.php?id=$1&p=$2 [L]
RewriteRule ^forum-([0-9]+)(.*)$ viewforum.php?id=$1 [L]
RewriteRule ^sujet-([0-9]+)-([0-9]+).html$ viewtopic.php?id=$1&p=$2 [L]
RewriteRule ^sujet-([0-9]+)(.*)$ viewtopic.php?id=$1 [L]
RewriteRule ^message-([0-9]+)(.*)$ viewtopic.php?pid=$1 [L]
RewriteRule ^profil-([0-9]+)(.*)$ profile.php?id=$1 [L]
…
```

Autrement dit, des règles assez imbuvables, mais surtout un système où il est impossible de rajouter des règles sans éditer ce fichier.

Imaginez que toutes les extensions de WordPress qui ont des besoins spécifiques rajoutent leur propre réécriture. On ne s'y retrouverait pas…

WordPress fonctionne différemment. Au lieu de réécrire les adresses directement depuis les fichiers .htaccess, il redirige toutes les requêtes sur le fichier index.php.

L'analyse des URL est donc faite directement depuis le code de WordPress. C'est un peu plus coûteux en performances, mais c'est beaucoup plus souple. De ce fait, il est très facile de changer la forme des permaliens depuis l'interface d'administration et il est facile d'étendre la réécriture *via* les extensions.

Le fichier .htaccess de WordPress

Le fichier .htaccess par défaut de WordPress ressemble à ceci :

```
# BEGIN WordPress
<IfModule mod_rewrite.c>
RewriteEngine On
RewriteBase /
RewriteCond %{REQUEST_FILENAME} !-f
RewriteCond %{REQUEST_FILENAME} !-d
RewriteRule . /index.php
</IfModule>
# END WordPress
```

Il peut être un peu différent si WordPress est installé dans un sous-répertoire.

Cela se passe de la façon suivante :

- On vérifie la présence du module rewrite d'Apache 2.

- On active le moteur de rewriting.
- On définit la base du site : / pour un site WordPress installé à la racine du domaine.
- On vérifie que la requête HTTP ne porte pas sur un fichier qui existe réellement. S'il n'existe pas, aucune règle n'est appliquée, sinon on redirige sur le fichier index.php.

Les fichiers .htaccess ne sont valables que pour les serveurs HTTP tels qu'Apache version 1 et 2 ou le serveur alternatif LiteSpeed.

Si vous utilisez un serveur HTTP de nouvelle génération comme Lighttpd ou Nginx, la logique de réécriture est la même, à la différence que les règles sont à ajouter directement dans le VHOST du fichier de configuration du serveur.

Depuis la version 2.8 de WordPress, la réécriture d'URL fonctionne également sur le serveur IIS de Microsoft.

Fonctions

add_rewrite_rule($regex, $redirect, $after = 'bottom')

Cette première fonction permet d'ajouter une réécriture d'URL dans WordPress. Pour bien comprendre le fonctionnement, nous allons prendre comme exemple une extension de calendrier.

L'objectif est de réécrire des adresses de la forme : *http://monsite.fr/moncalendrier/2008/06*.

Pour cela, la fonction `add_rewrite_rule()` prend trois paramètres :

- L'expression régulière qui va cibler les données à récupérer dans l'URL.
- L'adresse sur laquelle nous allons rediriger les données, cette dernière étant utilisée uniquement en interne dans WordPress.
- Le dernier paramètre permet de spécifier si la règle est exécutée avant les règles de WordPress ou après. Ce dernier paramètre n'est utile que dans de très rares cas.

Ici nous allons utiliser la fonction avec les paramètres suivants :

```
add_rewrite_rule('(moncalendrier)/[/]?([0-9]*)[/]?([0-9]*)$',
'index.php?pagename=$matches[1]&var1=$matches[2]&var2=$matches[3]');
```

Le premier paramètre permet de cibler les deux parties de la date : 2008 et 06.

Le deuxième paramètre construit l'adresse réelle, la variable `$matches` contient les données récupérées par l'expression régulière.

- `$matches[1]` contient moncalendrier.
- `$matches[2]` contient 2008.
- `$matches[3]` contient 06.

Petite précision, il n'est pas obligatoire de rediriger la requête sur une page en particulier. Vous pouvez très bien garder la page index.php, passer uniquement les deux paramètres liés à la date, tout en sachant que vous pourrez gérer le fichier du thème à afficher dans le code de l'extension (tout cela est décortiqué au Chapitre 11).

add_rewrite_tag($tagname, $regex)

Cette fonction permet de rajouter des marqueurs dans les permaliens étant au format personnalisé de WordPress. Les marqueurs que vous pouvez ajouter sont semblables à ceux que vous trouvez dans la page permaliens de l'interface d'administration, comme `%postname%`, l'identifiant de l'article.

La liste complète des marqueurs présents par défaut est disponible à l'adresse suivante : **http://codex.wordpress.org/Using_Permalinks#Structure_Tags**.

La fonction prend deux paramètres : le premier est le nom du tag entouré du caractère %, et le second paramètre est une expression régulière pour définir quels sont les caractères autorisés dans le tag.

Par exemple, pour conserver les chiffres et les lettres, vous utiliserez le méta caractère `(.+)`.



Cette fonction doit être utilisée avant ou durant l'action `init` pour être prise en compte par WordPress.

add_feed($feedname, $function)

Cette fonction permet d'ajouter un type de flux de données. Par défaut, WordPress peut afficher des flux de données au format RSS1, RSS2 et Atom.

Rien ne vous empêche de créer votre propre format pour vos besoins personnels.

Voici les adresses des flux de données d'un site WordPress :

- RSS2 (défaut) : *http://monsite.fr/feed/* ou *http://monsite.fr/feed/rss2/;*
- RSS1 : *http://monsite.fr/feed/rss1/;*
- Atom : *http://monsite.fr/feed/atom/.*

Cette fonction prend comme premier paramètre le nom du flux de données (rss3 par exemple), tandis que le second paramètre peut recevoir le nom de la fonction à exécuter ou le chemin d'un fichier PHP à exécuter.

add_rewrite_endpoint($name, $places)

Cette fonction permet d'ajouter un mot-clef en fin d'adresse. Par exemple, par défaut dans WordPress, les articles possèdent un endpoint pour l'utilisation des trackbacks.

Cela permet de créer une adresse spécifique, par exemple :

- adresse de départ : *http://monsite.fr/monarticle/;*
- adresse avec le endpoint : *http://monsite.fr/monarticle/trackback/.*

Le premier paramètre est le nom du mot-clef de fin ; le second argument est un tableau PHP contenant les constantes prédéfinies de WordPress.

Ces constantes représentent certains types de vues. Il faut passer en paramètre le type de vue où l'on souhaite utiliser le mot-clef de fin.

Cette liste de constantes PHP est disponible dans le code source du fichier wp-includes/rewrite.php.

url_to_postid($url)

Cette fonction ne permet pas de modifier ou d'étendre le rewriting de WordPress. Elle permet simplement de récupérer l'ID d'un article depuis son URL.

Elle prend comme seul paramètre l'adresse Internet et vous retourne l'ID de l'article s'il existe ou le chiffre 0 s'il n'existe pas.

Gestion des JavaScripts – *WP_Scripts*

Concept

WordPress est agrémenté de différentes bibliothèques JavaScript. Ces dernières permettent les traitements Ajax de l'interface d'administration ou encore l'éditeur visuel de WordPress.

Pour gérer les JavaScript, les développeurs de WordPress ne placent pas manuellement les balises HTML des scripts sur chaque page selon les besoins.

Ils ont mis en place un mécanisme de dépendance, d'enregistrement et de traduction des bibliothèques JavaScript. Ce mécanisme fonctionne *via* la classe `WP_Scripts`.

Par exemple, la page de rédaction de WordPress a besoin d'une dizaine de fonctions et de quatre bibliothèques JavaScript que sont Prototype, jQuery, Scriptaculous et TinyMCE.

L'ordre d'écriture de ces scripts, évitant les conflits et autres problèmes, est géré dynamiquement par la classe `WP_Scripts`.

Fonctions

wp_print_scripts($handles = false)

Cette fonction est automatiquement appelée dans l'en-tête HTML de WordPress, que ce soit dans l'interface d'administration ou dans le thème du site.

Elle se charge d'écrire le code HTML des JavaScript chargés dans WordPress, de façon qu'il n'y ait pas d'erreurs de dépendance.

Depuis WordPress 2.8, cette fonction dispose des équivalents spécifiques suivants :

- `print_head_scripts()` : pour l'en-tête de l'administration ;
- `print_footer_scripts()` : pour le pied de page de l'administration ;
- `wp_print_head_scripts()` : pour l'en-tête du thème ;
- `wp_print_footer_scripts()` : pour le pied de page du thème.

wp_enqueue_script($handle, $src = false, $deps = array(), $ver = false, $in_footer = false)

Cette fonction permet de charger un script dans WordPress. Elle est utilisée généralement pour inclure un script déjà présent de WordPress.

Par exemple, WordPress contient la librairie jQuery. Pour appeler jQuery uniquement sur la vue détaillée d'un article, nous placerons le code suivant :

```
if ( is_single() ) {
  wp_enqueue_script('jquery');
}
```

Il est possible de compléter les autres paramètres de la fonction, mais la fonction `wp_register_script()` sert généralement pour un usage plus poussé.

wp_register_script($handle, $src, $deps = array(), $ver = false, $in_footer = false)

Tout comme `wp_enqueue_script()`, cette fonction permet de charger un script dans WordPress.

Cependant, cette fonction est toujours utilisée pour charger un script tiers à WordPress, car le paramètre `$src` est obligatoire.

Par exemple, pour charger une extension jQuery permettant de remplacer les sons de votre contenu par un lecteur MP3, vous utiliserez le code suivant :

```
wp_register_script( 'jquery-sons', 'http://monsite.fr/wp-content/jquery-
plugins/son.js', array('jquery'), '1.0' );
```

La fonction prend quatre paramètres :

- **L'identifiant du JavaScript.** Il doit être unique ; s'il existe déjà, WordPress n'enregistrera pas votre script.
- **L'adresse du fichier JavaScript.** Elle sera utilisée par l'attribut SRC dans le code HTML.
- **Le tableau des dépendances.** Par exemple, ici nous utilisons une extension jQuery ; cela veut dire que jQuery est nécessaire au bon fonctionnement. Nous ajoutons donc l'identifiant `jquery` dans le tableau des dépendances. WordPress se chargera automatiquement de lancer jQuery si cela est nécessaire.
- **Le numéro de version.** Utile pour mettre à jour le JavaScript et éviter par la même occasion les problèmes de cache. Il suffit de mettre le numéro de version du script que vous utilisez et le tour est joué.

wp_deregister_script($handle)

Cette dernière fonction permet de décharger un JavaScript dans WordPress. Ce qui peut être le cas si vous souhaitez lancer une version différente que les librairies incluses dans WordPress.

Elle prend comme seul paramètre l'identifiant du JavaScript à désactiver.

Bonnes pratiques

Pour connaître la liste des librairies incluses et leur identifiant, vous pouvez consulter la page du Codex de la fonction `wp_enqueue_script()` : **http://codex.wordpress.org/Function_Reference/wp_enqueue_script#Parameters**.

Les principales librairies sont :

- Prototype ;
- Scriptaculous ;
- jQuery et jQuery UI ;
- SWFObject ;
- ThickBox ;
- TinyMCE.

Notez que l'objectif des développeurs de WordPress est de conserver à terme uniquement le framework jQuery, pour des raisons évidentes de performances et de cohérence dans le code.

Pour insérer du JavaScript dans l'interface d'administration sans passer par la classe `WP_Scripts`, il suffit d'ajouter une action sur l'événement `admin_print_scripts`.

Enfin, les fonctions de `WP_Scripts` doivent être utilisées au plus tard lors de l'événement `template_redirect`, l'idéal étant `init` ou `admin_init` selon le cas.

La classe cron de WordPress – *WP_Cron*

C'est quoi un cron ?

Avant de parler du concept dans WordPress et des fonctions, voici un petit rappel de ce qu'est un cron.

cron est le nom d'un programme qui permet aux utilisateurs des systèmes Unix, principalement Linux et Mac OS X, d'exécuter automatiquement une action spécifique à une date et une heure précisées à l'avance, ou selon un cycle défini à l'avance.

Il s'agit d'une fonctionnalité très utile pour des tâches routinières d'administration système, mais elle peut très bien être exploitée pour tout autre chose. Par exemple, vous pouvez demander à cron de jouer tel fichier MP3 tous les jours à 7 heures sauf le samedi et le dimanche, afin de vous réveiller en musique.

Source : **http://fr.wikipedia.org/wiki/Cron**.

Implémentation dans WordPress

Malheureusement, cron est un outil accessible uniquement *via* SSH. De ce fait, il n'est pas accessible dans les hébergements mutualisés.

Pour permettre d'automatiser des actions quel que soit le type d'hébergement de WordPress, l'équipe de développeurs a intégré dans WordPress une classe PHP qui reproduit les fonctionnalités du logiciel cron, mais qui est complètement indépendante vis-à-vis du système.

Cette classe est d'abord apparue sous la forme d'une extension, avant d'être intégrée dans le cœur de WordPress 2.1. Un usage courant de `WP_Cron` est la sauvegarde automatique et l'envoi par e-mail de la base de données MySQL. Cette fonctionnalité est possible *via* différentes extensions.

Fonctions

wp_schedule_single_event($timestamp, $hook, $args = array())

Cette fonction permet de programmer une action qui s'exécutera une seule fois dans le temps.

Elle prend trois arguments :

- la date d'exécution de l'action (au format Timestamp Unix) ;
- la fonction PHP à exécuter lors de l'action ;
- un tableau PHP contenant les paramètres passés à la fonction exécutée.

wp_schedule_event($timestamp, $recurrence, $hook, $args = array())

Cette fonction permet de programmer une action qui se répétera indéfiniment dans le temps.

Elle prend un paramètre en plus que la fonction `wp_schedule_single_event()` ; il est positionné en deuxième position et il permet de choisir le format de récurrence.

Par défaut, ce paramètre accepte les récurrences suivantes :

- hourly, soit toutes les heures ;
- daily, soit tous les jours.

Il est possible d'ajouter des récurrences *via* des extensions WordPress.

Le premier paramètre, `$timestamp`, permettra de préciser cette fois-ci la date de première exécution de l'événement.

wp_unschedule_event($timestamp, $hook, $args = array())

Cette fonction permet de supprimer la programmation d'une action dans WordPress. Pour pouvoir l'utiliser, il est nécessaire de connaître la date de prochaine exécution de l'événement, la fonction appelée et les arguments que prend la fonction.

Ces informations sont à passer dans les trois paramètres que prend la fonction.

wp_clear_scheduled_hook($hook, $args = array())

Cette fonction permet de supprimer toutes les programmations depuis le hook passé en paramètre. Cela veut dire que si deux programmations exécutent la même fonction (hook) à des intervalles différents, les deux seront supprimées.

wp_next_scheduled($hook, $args = array())

Enfin, cette dernière méthode permet de retourner la date de prochaine exécution d'une action. Pour cela, il est nécessaire de passer deux paramètres :

- la fonction utilisée par l'action (hook) ;
- les arguments que prend la fonction exécutée.

Bonnes pratiques

La classe `WP_Cron` peut être très utile pour des programmations basiques n'ayant pas besoin de précision dans l'heure d'exécution.

En effet, imaginez que vous programmiez une action tous les jours à 17 heures. Si pendant une journée aucun visiteur ne consulte votre site, aucune exécution PHP du logiciel WordPress n'a lieu. De ce fait, aucune exécution ne sera faite à 17 heures.

Autre différence notable avec une programmation cron classique, c'est la façon dont est exécuté le PHP. Généralement, l'envoi d'une newsletter est automatisé sur un serveur à des heures matinales (4-5 heures du matin). Cela veut dire que cron va exécuter un processus CLI de PHP, sans passer par un serveur web.

Sans entrer plus dans les détails, disons que l'utilisation d'un processus CLI permet une exécution directe de PHP, ce qui améliore les performances et résout pas mal de problématiques, surtout lors de traitements processeur massifs et intensifs, comme l'envoi multiple d'e-mails.

Bref, il est important de faire attention à l'usage que vous faites de `WP_Cron`.

Fonctions de formatage

Concept

WordPress possède différentes fonctions de formatage. Elles permettent de formater, corriger, sécuriser les chaînes de caractères selon le contexte et le comportement attendu.

Il existe une cinquantaine de fonctions propres au formatage des données dans WordPress ; nous ne présenterons que les plus utilisées.

Les fonctions *esc_*()*

Dans la section de ce chapitre consacrée aux fonctions d'internationalisation de WordPress, nous avons plusieurs fonctions de type `esc_*()`. Il en existe d'autres, plus généralistes.

Leur principe reste le même que celui de la fonction PHP `specialchars()` : convertir (ou échapper) les caractères spéciaux d'une chaîne pour les rendre valide dans le contexte donné.

À l'origine, les développeurs de WordPress avaient conçu `wp_specialchars()` pour cet usage, mais elle a été rendue obsolète depuis la version 2.8, et son fonctionnement a été précisé au sein des fonctions suivantes, avec des noms normalisés :

- `esc_html($text)` : utilisation au sein d'un bloc HTML. Remplace `wp_specialchars()` ;
- `esc_attr($text)` : utilisation au sein d'un attribut HTML. Remplace `attribute_escape()` ;
- `esc_js($text)` : utilisation au sein d'un affichage JavaScript. Remplace `js_escape()` ;
- `esc_sql($sql)` : une utilisation au sein d'une requête SQL (équivalent de `$wpdb->escape()`) ;
- `esc_url($url, $protocols = null)` : vérifie et nettoie une adresse web. Remplace `clean_url()` ;
- `esc_url_raw($url, $protocols = null)` : idem, mais à destination de la base de données. Remplace `sanitize_url()`.

wp_specialchars_decode($string, $quote_style = ENT_NOQUOTES)

Cette fonction fait l'inverse de `esc_html()` (et donc de l'ancienne fonction `wp_specialchars()`) : elle transforme une entité HTML en son caractère spécial correspondant. Elle sert surtout pour les caractères &, <, >, " et '.

wptexturize($text)

Cette fonction permet de convertir une liste de mots et de caractères en leur entité HTML. Ces modifications sont très légères et affectent peu de mots de la langue française.

Par exemple, deux tirets à la suite seront remplacés par l'entité HTML `—`.

Le seul paramètre est la chaîne de caractères à filtrer.

wpautop($pee, $br = 1)

Cette fonction permet de remplacer les doubles retours à la ligne de la chaîne de caractères par des paragraphes HTML (`<p>…</p>`).

WordPress utilise cette fonction pour formater le contenu des articles ainsi que leur extrait.

La fonction prend deux paramètres : le premier est la chaîne de caractères à filtrer, le second prend un booléen, qui préserve ou non les retours à la ligne `<br />` dans le code HTML.

force_balance_tags($text)

Cette fonction permet de contrôler la conformité du code HTML. Pour cela, la fonction vérifie que chaque balise HTML est bien ouverte et bien fermée.

Cette fonction transformera le code HTML malformé suivant :

```
<p><a href="">mon lien</p>
```

en un code valide :

```
<p><a href="#">mon lien</a></p>
```

is_email($user_email)

Cette fonction permet de tester si la chaîne passée en paramètre correspond ou non à une adresse e-mail.

Si c'est une adresse e-mail valide, la valeur de retour sera `true`, sinon ce sera `false`.

sanitize_email($email)

Cette fonction permet de nettoyer une adresse e-mail. Pour cela, elle se contente de supprimer tous les caractères non autorisés par le format des adresses e-mail.

make_clickable($ret)

Cette fonction permet de remplacer les adresses Internet d'un texte par des liens. Par exemple, quand un utilisateur tape **http://google.fr** dans un commentaire, un lien vers Google sera automatiquement ajouté sur ce texte !

remove_accents($string)

Cette fonction permet de supprimer les accents d'une chaîne de caractères. Elle prend comme seul paramètre la chaîne de caractères à filtrer.

sanitize_title($title, $fallback_title = '')

Cette fonction permet de nettoyer les chaînes de caractères des caractères spéciaux. Elle transforme toute la chaîne en minuscules et ne garde que les caractères de l'alphabet et les chiffres, autrement dit les caractères compatibles d'une adresse Internet. Enfin les espaces sont remplacés par des tirets.

Ce qui donne :

- **Chaîne de départ.** Et voilà un super titre pour cet édito ! N'est-ce pas ?
- **Chaîne nettoyée.** et-voila-un-super-titre-pour-cet-edito-nest-ce-pas

Cette fonction est utilisée pour construire les slugs dans WordPress, ou identifiants de l'article en français. Cette information est nécessaire à la construction des adresses Internet des articles.

sanitize_user($username, $strict = false)

Cette fonction permet de formater les identifiants des utilisateurs de WordPress lors de leur création. Cela évite les caractères spéciaux et autres entités HTML.

Elle prend deux paramètres : le premier est la chaîne de caractères du nom d'utilisateur, le second paramètre permet de filtrer un peu plus l'identifiant, en limitant les caractères autorisés aux caractères ASCII pour un maximum de compatibilité.

sanitize_file_name($name)

Cette fonction fait la même chose que la fonction `sanitize_title()` à la différence qu'elle conserve les points dans la chaîne de caractères.

Elle est utilisée pour formater les noms de fichiers lors de l'envoi d'un média dans WordPress.

sanitize_html_class($class, $fallback = '')

Cette fonction permet de s'assurer qu'un nom de classe HTML ne comprend que des caractères valides. Le second argument permet de préciser quelle chaîne renvoyer s'il se trouve que la chaîne nettoyée soit vide...

sanitize_text_field($string)

Cette fonction permet de nettoyer les chaînes en provenance de l'utilisateur ou de la base de données. Elle vérifie que les caractères sont tous en UTF-8, convertit les caractères spéciaux en entités, enlève les balises HTML, les retours à la ligne, les tabulations et les espace en trop, et enfin enlève les octets indésirables.

wp_strip_all_tags($string, $remove_breaks = false)

Cette fonction enlève toutes les balises HTML de la chaîne, notamment `<script>` et `<style>`.

wp_check_invalid_utf8($string, $strip = false)

Cette fonction vérifie que tous les caractères d'une chaîne sont bien encodés en UTF-8. Le second argument précise s'il faut tenter de supprimer les caractères non valides ou non.

translate_smiley($smiley)

Cette fonction permet de transformer un smiley en l'image équivalente, dans une balise `<img>`.

shortcode_unautop($content)

Cette fonction s'assure que les shortcodes ne sont pas encadrés de balises `<p>...</p>`.

capital_P_dangit($text)

Cette fonction est plus un clin d'œil qu'autre chose : elle repère les chaînes contenant "Wordpress" écrit avec un P minuscule et passe celui-ci en majuscule afin qu'il soit écrit correctement : "WordPress".

KSES – le filtre HTML de WordPress

Les critiques concernant le filtre HTML de WordPress sont nombreuses. En effet, il n'est pas rare de voir une balise ou un attribut HTML effacé automatiquement par WordPress.

Le responsable, c'est la bibliothèque de filtre KSES. Cette bibliothèque permet de définir les balises HTML ainsi que leurs attributs autorisés par WordPress.

Vous pouvez visionner ce tableau dans le fichier wp-includes/kses.php de WordPress. Ce fichier contient deux tableaux de filtre : un premier assez complet destiné aux articles de WordPress, et un second plus basique destiné aux commentaires.

Les tableaux se présentent sous la forme suivante :

```
$allowedposttags = array(
  'a' => array(
      'class' => array (),
      'href' => array (),
      'id' => array (),
      'title' => array (),
      'rel' => array (),
      'rev' => array (),
      'name' => array (),
      'target' => array()),
  …
```

Nous pouvons conclure que ce tableau autorise la balise HTML `a`, permettant de créer des liens. WordPress autorise les attributs `class`, `href`, `id`, `title`, `rel`, `rev`, `name` et `target`.

Si vous essayez un autre attribut, comme `lang`, il sera automatiquement supprimé par WordPress.

Enfin, il faut savoir que le contenu des articles de WordPress est filtré par Kses *via* un filtre appliqué sur `the_content`.

wp_kses($string, $allowed_html, $allowed_protocols)

Cette fonction permet d'exécuter le filtre Kses sur une chaîne de caractères. Elle prend trois paramètres : la chaîne de caractères à filtrer, le tableau des éléments HTML autorisés (comme nous l'avons vu ci-dessus) et le tableau des protocoles autorisés.

Pour le deuxième paramètre, il est possible d'utiliser le tableau par défaut des articles de WordPress *via* la variable globale `$allowedposttags`.

Le tableau par défaut des protocoles autorisés est : HTTP, HTTPs, ftp, mailto, news, irc, gopher, nntp, feed et telnet.

Fonctions de date

Principe

Les dates sont toujours stockées de la même façon dans WordPress. L'équipe des développeurs a choisi d'utiliser le format `DATETIME` de la base de données MySQL.

Ce format se présente sous la forme suivante :

```
YYYY-MM-DD HH:MM:SS
```

Par exemple, la date 22 juin 2008 à 16h42 et 13 secondes donnera 2008-06-22 16:42:13.

Fonctions

mysql2date($dateformatstring, $mysqlstring, $translate = true)

Cette première fonction permet de passer une date du format SQL `DATETIME` au format que vous souhaitez. Elle fonctionne de la même façon que la fonction `date()` de PHP pour définir le format de la date. N'hésitez pas à consulter la documentation de PHP, **http://fr2.php.net/date**, pour connaître les possibilités.

Elle prend trois paramètres :

- le format date désiré ;
- la date à convertir au format SQL ;
- un booléen pour autoriser ou non la traduction de la date.

date_i18n($dateformatstring, $unixtimestamp)

Cette fonction retourne une date traduite dans la langue du site WordPress. Le format de sortie est défini par le premier paramètre. Celui-ci prend les mêmes valeurs que la fonction `date()` de PHP.

Le second paramètre est le `TIMESTAMP` Unix de départ que l'on souhaite transformer.

current_time($type, $gmt = 0)

Cette dernière fonction permet de générer la date courante. Elle prend deux paramètres : le format de la date, et la prise en compte ou non du fuseau horaire lors de la génération de la date.

Le premier paramètre accepte deux valeurs :

- mysql génère une date au format DATETIME.
- timestamp génère une date au format Unix TIMESTAMP.

wp_timezone_supported()

Cette fonction renvoie les zones horaires reconnues par PHP.

Fonctions diverses

Les fonctions décrites ci-après sont difficiles à classer, car elles ne font pas partie d'une API particulière. Cependant, étant donné qu'on les retrouve très souvent dans le code de WordPress, leur description se justifie pleinement ici.

wp_die($message, $title = '')

Cette fonction est un dérivé de la fonction `die()` de PHP. Son objectif est de couper un script PHP avec un message d'erreur plus esthétique que la fonction classique de PHP.

Elle est généralement utilisée lorsqu'une condition de sécurité n'est pas remplie ou lors d'une mauvaise action dans l'interface d'administration de WordPress.

La fonction prend deux paramètres : le premier est le message à afficher dans le corps de la page et le second est le titre HTML de la page (la balise `title` de l'en-tête HTML).

wp_mail($to, $subject, $message, $headers = '', $attachments = array())

Cette fonction permet d'envoyer un e-mail. WordPress propose sa propre fonction pour l'envoi, qui n'utilise pas directement la fonction `mail()` de PHP, mais qui exploite la classe `PHPMailer`.

Cette fonction est pluggable ; cela veut dire qu'elle peut être remplacée par une fonction portant le même nom dans une extension.

Elle prend cinq paramètres :

- L'adresse e-mail du destinataire.
- Le sujet de l'e-mail.
- Le contenu de l'e-mail.
- Les deux derniers paramètres sont optionnels ; on peut y modifier l'en-tête de l'e-mail, et préciser les fichiers à joindre au message.

Cette fonction contient de nombreux filtres pour modifier son comportement.

wp_redirect($location, $status = 302)

Cette fonction permet de gérer correctement les redirections dans WordPress.

Au lieu d'utiliser la fonction `header()` de PHP plusieurs fois à la suite, WordPress propose une fonction aboutie pour gérer correctement les redirections quel que soit le type de serveur HTTP. Le serveur IIS de Microsoft ne gère pas les mêmes redirections que les serveurs HTTP présents sous Linux.

La fonction prend deux paramètres : l'adresse de destination et le code HTTP.

paginate_links($args = '')

Cette fonction permet de créer une pagination très rapidement. Elle est utilisée dans l'interface d'administration pour gérer la pagination du contenu.

Elle prend comme seul paramètre un tableau de données qui contient les informations suivantes :

- la base des URL à construire ;
- le format de la pagination qui correspond au mot-clef utilisé dans l'adresse ;
- le nombre total de pages ;
- la page courante, etc.

API Post Meta – Les métadonnées des articles

Concept

L'API des Post Meta, ou métadonnées des articles, met en place un système de stockage complètement générique pour stocker des données liées aux articles, aux pages et aux attachements de WordPress (*via* le gestionnaire de médias).

Cette API est très appréciée des développeurs d'extensions, car on peut y stocker tout type de contenu lié aux articles – comme un compteur de vue, la notation des articles, etc. – et cela de façon très simple, tout en profitant des fonctionnalités natives de cache de WordPress.

Néanmoins, il faut faire attention à l'usage de cette API. Le fait que la structure de la table soit complètement générique peut poser des problèmes de performance pour les extensions stockant un nombre important d'informations et réalisant beaucoup d'appels. Dans de tels cas, la création d'une table spécifique à l'extension peut être une excellente alternative, bien qu'elle demande un peu plus de développement.

Fonctions

add_post_meta($post_id, $meta_key, $meta_value, $unique)

Cette fonction permet d'ajouter une métadonnée à un article. Elle prend quatre paramètres :

- L'ID de l'article.
- L'identifiant de la donnée à stocker.
- La valeur à stocker.
- Un quatrième paramètre qui permet de préciser si la valeur à stocker est unique, ou bien si elle est multiple ; dans ce dernier cas, on enregistre un tableau sérialisé dans la base de données.

update_post_meta($post_id, $meta_key, $meta_value, $prev_value)

Cette fonction permet de mettre à jour une métadonnée. Pour cela, la fonction prend quatre paramètres :

- L'ID de l'article.
- L'identifiant de la valeur à stocker.
- La nouvelle valeur.

- Un dernier paramètre, optionnel, qui permet éventuellement de préciser l'ancienne valeur. Pour que la métadonnée soit mise à jour, il faudra que l'ancienne valeur corresponde bien à la base de données.

delete_post_meta($post_id, $key, $value)

Cette fonction permet de supprimer une métadonnée de la base de données. La fonction prend trois paramètres :

- L'ID de l'article.
- L'identifiant de la métadonnée à supprimer.
- Éventuellement, la valeur actuelle de la métadonnée. Si le dernier paramètre est fourni, la valeur de la métadonnée sera également vérifiée avant de procéder à la suppression.

get_post_meta($post_id, $key, $single)

Cette fonction permet de récupérer la valeur d'une métadonnée. Elle prend trois paramètres, dont les deux premiers sont obligatoires :

- L'ID de l'article.
- L'identifiant de la métadonnée.
- Un troisième paramètre qui permet de préciser par un booléen (`true` ou `false`) si la valeur est unique ou non. Si elle l'est, la valeur sera retournée directement ; sinon, la fonction retournera un tableau de valeurs.

get_post_custom($post_id)

Cette fonction retourne un tableau associatif avec l'ensemble des identifiants et des valeurs des métadonnées. Elle prend comme seul paramètre l'ID de l'article.

Si le paramètre n'est pas spécifié et que l'on se trouve dans la boucle de WordPress, la fonction tentera de récupérer l'ID de l'article. Si aucun ID n'est trouvé, la fonction retournera un booléen `false`.

get_post_custom_keys($post_id)

Cette fonction, tout comme `get_post_custom()`, prend comme seul paramètre l'ID de l'article. Ce dernier est optionnel et fonctionne de la même façon que la fonction précédemment notée. Elle permet de retourner le tableau des identifiants de métadonnées existants pour l'article.

get_post_custom_values($key, $post_id)

Cette dernière fonction permet de retourner le tableau des valeurs pour un identifiant de métadonnée donné. Elle retourne donc un tableau classique. Elle prend deux paramètres :

- l'identifiant de la métadonnée ;
- l'ID de l'article (optionnel).

Si aucun identifiant d'article n'est fourni, la fonction se comportera comme `get_post_custom()` et elle essayera de récupérer l'ID de l'article depuis la boucle.

API *WP_Query* – Requêtage de la base de données WordPress

Concept

Pour récupérer les articles et pages de la base de données, WordPress possède une API de requêtage très puissante. Cette API, nommée `WP_Query`, permet d'effectuer des requêtes sur la base de données de votre blog.

Lorsque l'on consulte une page de catégorie, la classe WP_Query va récupérer deux informations :

- le préfixe `category` ;
- le nom de la catégorie.

Grâce à ces deux informations, la classe va connaître le contexte dans lequel on se situe et construire la requête SQL nécessaire à la récupération des articles.

Ainsi, on obtient une classe complètement générique capable de récupérer des articles d'une catégorie, d'un tag ou encore d'une date. Cette classe apparaît très peu dans la partie thème de WordPress. On retrouve généralement les fonctions `query_posts()` ou `get_posts()` pour créer des boucles de contenus personnalisés.

Il faut savoir que ces deux fonctions utilisent la classe `WP_Query`. Historiquement, la fonction `get_posts()` utilisait sa propre logique et ses propres paramètres. Néanmoins, depuis la version 2.6 de WordPress, la fonction `get_posts()` permet uniquement de faire un appel de classe `WP_Query` – avec quelques petites différences tout de même – tandis que la fonction `query_posts()` n'est qu'un alias permettant de simplifier l'appel à `WP_Query` en automatisant l'instanciation de la classe PHP.

Les informations concernant l'instanciation `WP_Query` interne à WordPress (celle qui gère le passage des URL à une requête SQL) sont accessibles *via* la variable globale PHP `$wp_query`. La création de la requête SQL est réalisée juste après le marqueur `init`.

Il est intéressant de noter que c'est cette classe qui permet les *conditionnal tags*, ou "marqueurs de thème conditionnels", avec lesquels vous créez des conditions dans vos thèmes. Par exemple, pour vérifier que l'on se trouve sur la page d'accueil de son blog, on utilisera la fonction PHP `is_home()`.

Enfin, cette classe fournit également les fonctions propres à la boucle WordPress (`have_posts()`, `the_post()`, etc.). Ces fonctions sont des alias vers les méthodes de la classe `WP_Query`.



Pour davantage d'informations sur la classe `WP_Query` : **http://codex.wordpress.org/Function_Reference/WP_Query** (en anglais).

Fonctions

is_*()

On retrouve un très grand nombre de fonctions commençant par `is_`. Ces fonctions permettent de tester un contexte ou la présence d'un objet. Par exemple, pour savoir si l'on se trouve sur la page d'une catégorie, il suffit d'utiliser la fonction `is_category()`.

Ce type de fonction retournera toujours la valeur `true` si la condition est remplie et `false` dans le cas contraire.

get_posts() et query_posts()

Ces deux fonctions possèdent des nuances (que l'on peut obtenir *via* le Codex) concernant les paramètres pris en compte, mais elles proposent toutes les deux les mêmes résultats, soit un requêtage spécifique sur la base des articles de WordPress. Ces fonctions retournent généralement un tableau d'objet, ou le booléen `false` si aucun résultat n'est renvoyé par la base de données.

wp_reset_query()

Cette fonction permet de réinitialiser la variable global `$wp_query`.

Lorsqu'on utilise la fonction `query_posts()`, on écrase la variable `$wp_query` avec les résultats de la nouvelle requête. Pour restaurer les résultats de la requête initiale de WordPress, il suffit d'exécuter la fonction `wp_reset_query()`.

setup_postdata()

Lorsqu'on utilise la fonction `get_posts()`, on récupère un tableau d'articles.

Pour afficher le contenu de ce tableau, on utilise une boucle PHP comme `foreach` ou `while`. Pour chaque itération, on utilise alors les fonctions des thèmes de WordPress, comme `the_permalink()` pour afficher le lien de l'article ou `the_title()` pour afficher le titre.

Néanmoins, avec une simple boucle PHP, toutes les fonctions de thèmes de WordPress ne fonctionnent pas, car toutes les données de l'article ne sont pas chargées en mémoire. Pour y remédier, il faut charger toutes les données de l'article en mémoire, et pour cela utiliser la fonction `setup_postdata()`. Cette dernière prendra comme unique paramètre l'objet article fourni par la boucle. Ainsi, on peut utiliser l'ensemble des données de l'article.

Bonnes pratiques

La fonction conditionnelle `is_home()` est régulièrement utilisée par les créateurs de thèmes "Magazine", parfois à tort. Cette fonction permet de vérifier si on se trouve sur la page d'accueil. Jusque-là, rien d'extraordinaire... Mais, sur la page 2 de la page d'accueil, la fonction `is_home()` sera toujours valide, car on ne sera pas dans une page d'archives avec date ou catégories, mais dans une page qui suit la pagination de la page d'accueil.

Pour remédier à ce problème, il est préférable d'utiliser la fonction `is_front_page()` : cette dernière n'est validée que pour la page d'accueil et sans pagination. Autrement dit, elle répond parfaitement à ce contexte.

Une autre très mauvaise pratique est l'utilisation à tout va de la fonction `query_posts()`. En effet, dans la création de site avec des fonctionnalités et une apparence plus CMS que blog, il est courant de trouver des fonctions `query_posts()` en tête de fichiers pour récupérer des informations différentes de celles prévues par WordPress. Cela peut paraître anodin mais, dans un contexte de montée en charge et de qualité de développement, le fait de mettre une fonction `query_posts()` en tête de fichier force WordPress à recréer et à exécuter une nouvelle requête SQL.

De ce fait, sur une même page, on retrouvera deux fois la même fonction travaillant sur la base des articles, pour peu que ces requêtes soient complexes et que votre site soit consistant en données. Vous aurez alors sur votre serveur web un surplus de charges qui pourra poser problème. Il faut donc veiller à utiliser la fonction `query_posts()` à bon escient, et, si l'on souhaite modifier la requête de WordPress pour une catégorie spécifique, il suffit de faire appel au filtre présent dans la classe `WP_Query`.

Custom Post Types API – créer son propre type de contenu

De quoi s'agit-il ?

L'un des grands apports de WordPress 2.9 est l'ajout du support de types de contenus personnalisés, apport encore amélioré avec la version 3.0.

WordPress peut stocker et afficher un grand nombre de types de contenus. En interne, ils sont tous stockés dans la table wp_posts, mais sont différenciés par la colonne post_types.

Par défaut, WordPress 3.0 reconnaît cinq types de contenus : les articles, les pages, les fichiers joints, les révisions et les menus de navigation.

Fonctions

register_post_type($post_type, $args = array())

C'est le plus souvent la seule fonction dont vous aurez besoin pour déclarer votre type de contenu. La déclaration se fait en quelques lignes, appelées les plus souvent par une action :

```
add_action( 'init', 'create_post_type' );
function create_post_type() {
  register_post_type( 'mon_produit',
    array(
      'labels' => array(
        'name' => __( 'Produits' ),
        'singular_name' => __( 'Produit' )
      ),
      'public' => true,
```

```
    )
  );
}
```

Le premier argument est le seul obligatoire, il indique l'identifiant du type de contenu. Le second argument est facultatif mais primordial car il permet de redéfinir de nombreux aspects du type de contenu, notamment la manière dont il est présenté dans l'interface d'administration.

add_post_type_support($post_type, $feature)

Cette fonction permet d'indiquer quelles sont les fonctionnalités de l'interface exploitables par le type de contenu défini. Cela permet de préciser l'interface d'un type de contenu déjà créé. Par exemple, voici comment ajouter un champ Extrait aux Pages :

```
add_action('init', 'extraits_pour_pages');
function extraits_pour_pages'() {
  add_post_type_support( 'page', 'excerpt' );
 }
```

Les interfaces utilisables sont : `title`, `editor`, `author`, `thumbnail`, `excerpt`, `trackbacks`, `custom-fields`, `comments`, `revisions`, `pages-attributes`.

Le second argument peut être un tableau PHP, pour ajouter plusieurs fonctionnalités d'un coup.

remove_post_type_support($post_type, $feature)

Fonction miroir de la précédente, elle permet de retirer une fonctionnalité de l'administration. Par exemple, si l'on ne souhaite pas que l'utilisateur puisse ajouter un extrait à son article :

```
add_action('init', 'articles_sans_extraits');
function articles_sans_extraits'() {
  remove_post_type_support( 'post', 'excerpt' );
 }
```

Le second argument ne peut être qu'une chaîne.

post_type_supports($post_type, $feature)

Cette fonction vérifie qu'un type de contenu peut utiliser une fonctionne donnée.

post_type_exists($post_type)

Cette fonction permet de vérifier que l'identifiant de type de contenu n'est pas déjà utilisé.

get_post_type($the_post = false)

Cette fonction renvoie le type du contenu en cours, ou d'un contenu donné (*via* son ID).

get_post_types($args = array(), $output = 'names', $operator = 'and')

Cette fonction renvoie une liste des type de contenu actuellement définis.

Le premier argument permet de filtrer les types selon la manière dont ils ont été mis en place. Par exemple, si l'on ne souhaite récupérer que les types hiérarchisés :

```
$args = array(
  'hierarchical' => true
);
$post_types = get_post_types( $args );
  foreach ( $post_types as $post_type ) {
    echo '<p>' . $post_type . '</p>';
  }
```

Le deuxième argument précise si l'on veut récupérer simplement la liste des identifiants des types, ou bien direcement un objet PHP pour chaque type.

Bonnes pratiques

Il est probable que de plus en plus de thèmes et extensions vont se mettre à exploiter cette nouvelle possibilité de WordPress. De fait, il est important que les différents types de contenus ne se chevauchent pas, et cela commence avec l'identifiant utilisé lors de la création du type de contenu. La simplicité veut que le développeur choisisse directement "produit" ou "annonce", mais pour peu qu'un autre type cherche à utiliser le même identifiant, l'utilisateur final n'obtiendra pas ce qu'il attend. Pensez donc à préfixer les identifiants de vos types de contenu, par exemple avec vos initiales ou le nom de votre société : "xb_produit", "wpfr_annonce", etc.

11 Utiliser WordPress en tant que CMS

Introduction

La communauté WordPress s'est longtemps battue, et se bat encore, contre une perception erronée de cet outil : WordPress ne serait bon que pour les blogueurs et, si l'utilisateur veut un "vrai" site, il faut qu'il utilise un "vrai" CMS, comme Drupal, Joomla ou SPIP.

Vous l'aurez compris, l'idée selon laquelle WordPress ne serait bon qu'à faire des blogs est depuis longtemps fausse : WordPress est tout autant un vrai CMS que les autres, et peut servir pour tous les types de projets web.

Fondamentalement, parler de vrai et de faux CMS (*content management system*, système de gestion de contenu) est trompeur : à partir du moment où un outil permet de publier sur Internet, il gère du contenu, c'est donc un CMS à part entière. Partant de là, il existe des CMS effectivement spécialisés "blog" : le projet libre et francophone Dotclear (**http://fr.dotclear.org/**) en est un excellent exemple – d'ailleurs, son slogan a longtemps été "Dotclear ne fait que du blog, et il le fait bien".

WordPress est sorti du créneau "que pour les blogs" en 2005, avec la version 1.5 et l'introduction du concept de "pages", du contenu hiérarchisé ne dépendant pas de la chronologie du blog. Cette première étape permet déjà beaucoup de choses, mais WordPress continue sur cette lancée et, en 2007, la version 2.3 introduit la capacité de créer des taxinomies personnalisées, c'est-à-dire en gros de créer son propre jeu de catégories pour un type de contenu. Enfin, la 2.9 introduit une importante fonctionnalité : le support des types de contenu personnalisés. Ce sont ces trois aspects qui seront abordés dans ce chapitre.

D'autres aspects relatifs à la gestion avancée de contenu sont également à mentionner : les champs personnalisés (1.2, 2004), un éditeur WYS/WYG (2.0, 2005), la sauvegarde automatique (version 2.1, 2007), le gestionnaire de fichiers média (2.1 également), le correcteur orthographique (2.1 encore), la possibilité de configurer une page statique en page d'accueil plutôt que les derniers articles du blog (2.1 toujours), la sauvegarde des versions/révisions (2.6, 2008), le gestionnaire de menus (3.0, 2010)... et tant d'autres fonctionnalités dont le but final est de simplifier la vie de l'utilisateur.

En définitive, c'est le thème d'un site qui définit si ce site est un blog, un site de rencontre, une galerie de photos ou encore une base de données de livres sur la chirurgie obstétricienne plastique du bulbe rachidien. Tout ce qui se passe en coulisse tient de la gestion de contenu, et n'importe quel CMS, même fortement orienté blog, doit être considéré comme un "vrai CMS" pour peu que les développeurs de thèmes aient les coudées franches...

Les pages statiques

L'ajout du support des pages statiques dans la version 1.5 a été le premier pas de WordPress en dehors du "pur blog". Auparavant, WordPress ne pouvait gérer que des articles, liés à la chronologie du blog. Il fallait fortement bidouiller le fonctionnement de WordPress pour en tirer autre chose.

Une fois les pages apparues, il devenait possible de gérer du contenu dans un nouveau contexte : non seulement ce contenu était totalement extérieur à la chronologie du blog, mais elles pouvaient par ailleurs être liées entre elles ! Il devenait possible de gérer des pages ou sous-pages, et partant de là des sites entiers.

Mieux encore, personne n'était obligé d'utiliser les articles : il était tout à fait possible de ne se servir de WordPress que pour gérer des pages statiques, et de profiter ainsi d'un CMS éprouvé et simple d'usage, ainsi que de tous ses apports communautaires (thèmes, extensions, support...). Rapidement, de nombreux développeurs ont profité de cet apport pour ne plus proposer que des sites propulsés par WordPress, qu'ils aient besoin d'un blog ou non !

Comment en créer

Votre installation de WordPress dispose déjà, par défaut, d'un article et de son commentaire, et d'une page À propos. Si vous savez créer des articles, vous pouvez aussi facilement créer des pages, et lier les pages entre revient à assigner une page parente à toute nouvelle page créée. En bref, créer et gérer des pages dans WordPress, et les lier entre elles, est l'enfance de l'art. Et, si vous n'avez jamais essayé, nous vous enjoignons à le faire : ce n'est pas plus cher.

Comment les afficher

C'est ici que l'on constate que la différence entre un blog et un site "régulier" ne tient réellement qu'à son thème, son habillage, quel que soit le CMS utilisé. Tout dépend, somme toute, de l'emphase que met la feuille de style sur le contenu "non blog".

Prenons le thème par défaut de WordPress. Celui-ci fait la part blog à l'aspect blog : les articles sont clairement mis en avant, tandis que les pages sont reléguées dans le menu (voir Figure 11.01).

Vous aurez beau créer de nouvelles pages, celles-ci ne seront indiquées que dans le menu de navigation, et bien sûr à l'adresse de chacune. Pour inverser la tendance, il faut, au choix ou combiné :

- aller dans la page Réglages > Lecture et faire que la page d'accueil affiche une page statique plutôt que les derniers articles ;
- modifier le thème ;
- modifier le menu de navigation pour qu'il n'affiche que les pages voulues.

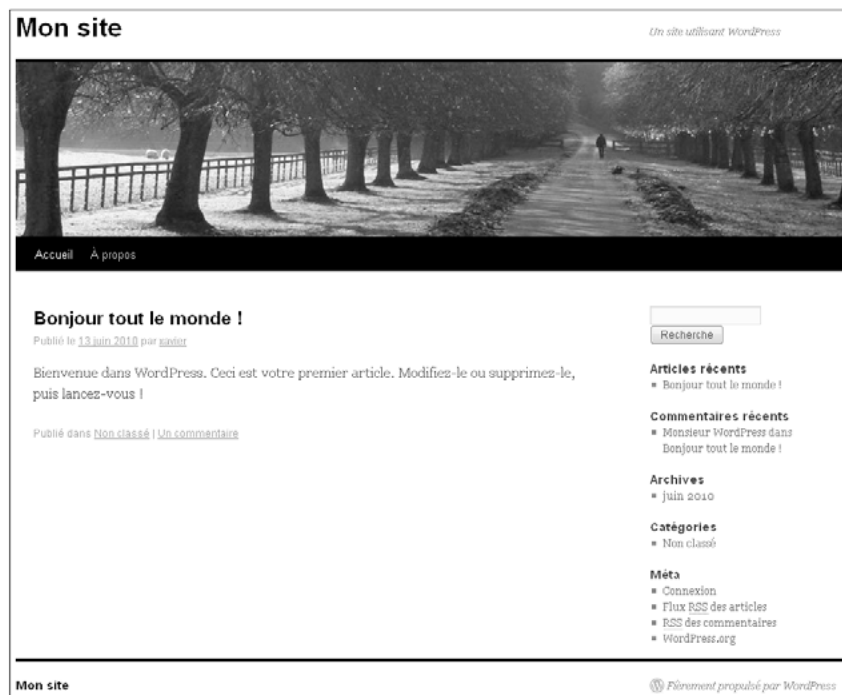


Figure 11.01

Twenty Ten, le thème par défaut de WordPress 3.0.

Comme vous l'avez vu aux chapitres traitant de la création d'un thème, l'affichage des articles se fait à partir de la Boucle (*The Loop*). À partir du moment où votre page d'accueil affiche une page statique plutôt que les articles, un premier pas est fait, la Boucle prend la suite. Le menu de navigation se charge ensuite de guider le visiteur au travers de pages du site.

Pour votre propre thème, le menu de navigation est la meilleure chose que vous puissiez ajouter si vous partez sur l'idée de faire un site plutôt qu'un blog. Le code à placer dans l'en-tête est très simple, tout en étant fortement personnalisable. Au plus simple, il suffit d'un appel de fonction qui produira la liste HTML nécessaire :

```
wp_nav_menu();
```

Partant de là, vous pouvez exploiter les diverses fonctions de `wp_nav_menu()` pour préciser vos besoins : déclarer la balise du contenant ainsi que sa classe et/ou son id, définir la profondeur des pages maximales, etc. Toutes ces options sont documentées sur le Codex : **http://codex.wordpress.org/Function_Reference/wp_nav_menu**.

Si vous souhaitez utiliser non pas le menu de navigation mais simplement les pages telles que vous les avez ordonnées dans l'administration de WordPress, vous pouvez faire appel à une autre fonction tout aussi puissante, et présente depuis la version 1.5 :

```
wp_list_pages();
```

Tout comme pour `wp_nav-menu()`, vous pouvez déclarer un tableau qui redéfinira le code HTML généré afin de l'adapter à vos besoins : profondeur, pages à exclure, affichage des sous-pages ou non, etc. Le Codex, en plus d'expliquer toutes ces options, offre de nombreux exemples pratiques pouvant servir d'inspiration : afficher les pages par date de création, ne lister que les sous-pages d'une page en particulier, etc. : **http://codex.wordpress.org/Function_Reference/wp_list_pages**.

Dans tous les cas, vous obtiendrez une liste de liens vers les contenus de votre site. Après, il ne tient qu'à vous de transformer ce code HTML en un menu à votre convenance, à l'aide des CSS, voire de JavaScript...

Les taxinomies personnalisées

Qu'est-ce qu'une taxinomie ?

Le sujet des taxinomies a déjà été abordé au chapitre précédent, sous l'angle purement développeur : présentation des faits, explication du contenu des tables, listage des fonctions de l'API.

L'objectif de cette section est de vous faire comprendre l'importance de cette API, et en quoi elle va vous aider à réaliser tous types de sites.

Revenons sur la définition de base : une taxinomie est une méthode de classification des informations. De fait, disposer d'une API de taxinomie signifie autoriser l'utilisateur (ou, du moins, le développeur d'extensions ou de thèmes) à créer ses propres classifications.

Avant sa version 2.3, WordPress ne disposait que d'une taxinomie : les catégories. Celles-ci s'appliquaient uniquement aux articles et aux liens. C'était déjà utile, car on pouvait créer des sous-catégories (on appelle cela une taxinomie hiérarchisée).

Avec l'arrivée des mots-clefs dans la version 2.3, l'intégralité de la gestion des taxinomies a été rendue générique. Désormais, les articles pouvaient accéder à une taxinomie hiérarchisée, les catégories, et également faire appel à une taxinomie non hiérarchisée, les mots-clefs. Chacune de ces taxinomies correspond à un usage précis, les catégories permettant un classement très général des articles sur site, tandis que les mots-clefs servaient aux petits détails qui ne s'appliquent pas forcément aux autres articles.

Ce n'est qu'avec la version 2.8 que l'API a évolué : en quelques lignes de code, il devenait possible d'ajouter ses propres taxinomies, WordPress prenant en charge la mise en place de l'interface sur la page d'écriture de l'article, ainsi que l'ajout automatique du menu dédié.

Mais les taxinomies ne fonctionnaient que pour les articles, et l'interface ne fonctionnait pas pour les taxinomies hiérarchisées.

WordPress 3.0 généralise réellement l'usage des taxinomies :

- On peut créer une taxinomie pour n'importe quel type de contenu (articles, pages, mais aussi les contenus personnalisés, comme nous le verrons dans la section suivante).
- L'interface d'administration fonctionne pour n'importe quel type de taxinomie (hiérarchique ou non).

- La gestion des taxinomies est grandement simplifiée (création, modification, suppression...).
- Il devient possible de préciser le texte des labels de l'interface, afin de mieux intégrer la taxinomie.

Ce que cela apporte

Vous pouvez donc mettre en place votre propre type de catégorisation.

Concrètement, cela signifie que vous pouvez avoir des groupements plus propres à votre contenu que "catégorie" ou "mot-clef".

Par exemple, si vous ne publiez sur votre site que des recettes de cuisine, peut-être voudriez-vous avoir, en plus des catégories et mots-clefs, un groupe nommé "ingrédients", un autre nommé "durée" et un dernier nommé "budget". Toutes ces choses sont réalisables avec les catégories et mots-clefs, mais seront beaucoup mieux intégrées à votre processus de travail (et à votre interface) si vous utilisez les taxinomies de base pour ce qu'elles sont, sans surcharger les catégories avec des centaines de termes qui n'ont aucun rapport entre eux (voir Figure 11.02).

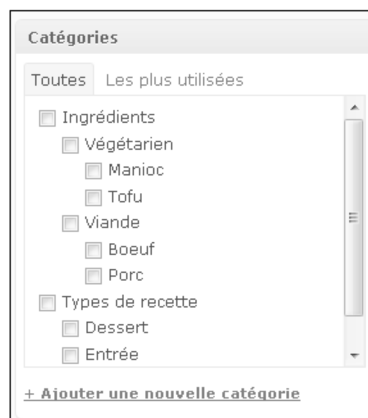


Figure 11.02
Avant : toutes les classifications sont mélangées dans les catégories.

En créant vos propres taxinomies, non seulement vous simplifiez la classification de votre contenu, mais vous faites un usage correct de WordPress, sans vous limiter à ses fonctionnalités de blog.

Comment en créer

L'ajout d'une nouvelle taxinomie ne requiert vraiment que l'utilisation de la fonction `register_taxinomy()`, déjà décrite succinctement au chapitre précédent.

Voici le code le plus simple pour ajouter une taxinomie d'ingrédients à vos articles :

```
register_taxonomy(
  'recipes_ingredients',
  'post',
  array(
    'hierarchical'    => true,
    'label'      => __('Ingredients'),
  )
);
```

Ce code est à placer soit dans une extension, soit dans le fichier functions.php de votre thème, selon le contexte de votre utilisation. La fonction est appelée *via* une action WordPress.

Dans l'ordre, nous définissons :

- **L'identifiant de la taxinomie.** Nous la préfixons car une autre extension pourrait définir sa propre taxinomie, comme "ingrédient" dans un contexte de boissons.
- **Le type de contenu.** Notre taxinomie ne s'applique ici qu'aux articles. Si l'on voulait l'appliquer aux articles, aux pages et au contenu personnalisé "Recettes", on pourrait utiliser un tableau : `array( 'post', 'pages', 'recipes' ),`.
- **Un tableau PHP définissant différentes options.** Ce tableau peut contenir de nombreuses options, mais dans cette version très simplifiée nous n'en utilisons que deux :
  - **hierarchical.** Précise si la taxinomie est hiérarchisée (comme les catégories) ou non (comme les mots-clefs).
  - **label.** Le nom qui s'affichera dans l'interface. Notez que nous utilisons la fonction d'internationalisation `__()` : nous faisons en sorte que notre code soit compréhensible du plus grand nombre, donc nous écrivons tous les textes en anglais (même l'identifiant), puis les rendons traduisibles (sauf, euh, l'identifiant).

À propos de texte et de compréhension : telle qu'elle est codée, notre taxinomie fonctionnera, mais l'interface présentera de nombreux défauts textuels. En effet, elle utilisera les mêmes chaînes que pour l'interface des catégories, comme Add New Category ou All Categories – bref, les valeurs par défaut des taxinomies de WordPress.

Pour résoudre ce problème, il nous faut reprendre entièrement les chaînes liées à la taxinomie. C'est le rôle de l'option labels, qui prend un tableau PHP :

```
register_taxonomy(
  'recipes_ingredients',
  'post',
  array(
    'hierarchical'        => true,
    'label'               => __('Ingredients'),
    'labels'              => array(
      'name'              => __('Ingredients'),
      'singular_name'     => __('Ingredient'),
      'search_items'      => __('Search Ingredients'),
      'popular_items'     => __('Popular Ingredients'),
      'all_items'         => __('All Ingredients'),
      'parent_item'       => __('Parent Ingredient'),
      'parent_item_colon' => __('Parent Ingredient:'),
```

```
            'edit_item'             => __('Edit Ingredient'),
            'update_item'           => __('Update Ingredient'),
            'add_new_item'          => __('Add New Ingredient'),
            'new_item_name'         => __('New Ingredient Name'),
            'separate_items_with_commas' => __('Separate ingredients with commas'),
            'add_or_remove_items'   => __('Add or remove ingredients'),
            'choose_from_most_used'  => __('Choose from the must used ingredients')
          )
        )
      );
```



Les trois dernières options ne sont vraiment utiles que pour les taxinomies non hiérarchisées.

Ici encore, nous avons internationalisé toutes les chaînes avec `__()`, pour garantir une diffusion globale de notre code. Les chaînes seront ensuite récupérées dans un fichier PO et traduites en français dans un fichier MO (notez que, par souci de place, nous avons omis le dernier argument, définissant le domaine de la traduction). Bien entendu, vous pouvez choisir d'utiliser directement des chaînes en français...

Une fois les chaînes bien en place, votre taxinomie s'intègre parfaitement au processus rédactionnel de votre article : WordPress prend en charge la création des pages de gestion (voir Figure 11.03), ainsi que les blocs dans la page de rédaction du type de contenu choisi (voir Figure 11.04). Il ne vous reste plus qu'à remplir chaque taxinomie avec les termes adéquats.

Figure 11.03
L'interface générée pour la taxinomie "Ingrédients".

Figure 11.04

Les interfaces autogénérées des trois taxinomies.

Bien d’autres options sont disponibles, mais pour commencer vous pouvez fonctionner avec leurs valeurs par défaut. Pour en savoir plus, comme toujours, il y a le Codex : **http://codex.wordpress.org/Function_Reference/register_taxonomy**.

Notez que cette page du Codex indique également la liste des identifiants réservés, c’est-à-dire les termes que vous ne pouvez pas utiliser pour qualifier vos taxinomies.

Comment les afficher

Maintenant que vous avez vos taxinomies, vous pouvez aller toujours plus loin dans la compartimentation de votre contenu. Mais il reste à le rendre exploitable par les visiteurs de votre site, sinon cela n’a vraiment d’intérêt que pour vous...

Nous allons voir deux manières d’y parvenir : le listage de termes, et le nuage de termes.

Vous pouvez lister les termes d’une taxinomie de l’article en cours simplement en utilisant la fonction `get_the_term_list()` :

```
<?php echo get_the_term_list( $post->ID, 'recipes_ingredients',
'Les ingrédients de cette recette ', ', ', ''); ?>
```

Dans le cas d’une taxinomie hiérarchisée, vous pouvez exploiter la fonction `wp_tag_cloud()`, simplement en modifiant la taxinomie prise en compte :

```
<?php wp_tag_cloud( array( 'taxonomy' => 'recipes_ingredients' ) ); ?>
```

Les types de contenu personnalisés

Qu'est-ce qu'un type de contenu ?

Un type de contenu est une appellation générique pour la manière dont sont stockées et gérées vos données par WordPress. Cela peut prendre n'importe quelle forme, et WordPress en reconnaît cinq par défaut :

- les articles ;
- les pages statiques ;
- les fichiers joints ;
- les révisions ;
- les menus de navigation (depuis la version 3.0).

Depuis le début, les utilisateurs de WordPress n'ont pas pu sortir de cette sémantique : s'ils voulaient mettre en place du contenu, il fallait l'enregistrer en tant qu'article ou que page. Ces deux types sont suffisamment génériques et dissociés pour répondre à la plupart des demandes mais, dans le cas de contenus spécifiques, le risque était grand de mélanger torchons et serviettes, pour ainsi dire.

L'ajout des *custom post types*, c'est-à-dire la possibilité de créer ses propres types de contenu, date de WordPress 2.9, mais l'implémentation est alors trop brute pour être vraiment utile : le développeur souhaitant créer ses propres types de contenu devait également bidouiller WordPress pour que celui-ci affiche une page d'administration ou prenne en compte les règles de réécriture. Les développeurs ont bien compris cela, et la version 3.0 automatise tous les processus laborieux, pour rendre la création de types de contenu très rapide.

Les types de contenu personnalisés (TCP) sont la pierre finale : ils font entrer WordPress dans la cour des grands CMS.

Ce que cela apporte

Vous êtes très heureux avec votre blog : des articles pour vos contenus quotidiens, des pages statiques pour votre contenu fixe (À propos et compagnie). Soit ! Vous n'avez alors pas besoin des TCP.

Mais l'usage des TCP transcende celui des blogs : avec eux, il s'agit réellement de gérer des contenus autres que des articles chronologiques ou des pages génériques.

Reprenons l'exemple de notre site de recette. Vous pouvez mettre les recettes dans des articles et votre contenu fixe dans des pages. Mais, si vous voulez également gérer un blog, où mettre vos articles ? Les mélanger avec les recettes ? Inversement, si vous utilisez les articles pour votre blog et les pages pour vos recettes, les pages sont-elles le meilleur endroit pour stocker également vos recettes. Non. Idéalement, il faudrait un troisième type de contenu : les recettes. C'est exactement ce qu'il est possible de faire depuis WordPress 3.0 : le CMS s'ouvre à toutes les formes de publication.

Qui plus est, WordPress prend tout en charge dès que le nouveau type est créé : interface d'administration, intégration à tous les niveaux, réécriture d'URL... Et vous pouvez créer autant de types de contenu que vous avez de contenus différents : recettes, critiques de livres de recettes, grands cuisiniers, critiques de restaurants... Tout cela en plus des articles et pages, qui conservent leur usage premier : la gestion du blog et des contenus fixes.

Comment en créer

Tout comme les taxinomies, l'ajout du nouveau type de contenu revient à ajouter une ligne dans une extension ou dans le fichier functions.php du thème :

```
register_post-type( 'recipes' );
```

C'est tout. C'est le code le plus simple pour obtenir un nouveau type de contenu : indiquer simplement l'identifiant. Bien sûr, de nombreuses options existent, et il est même primordial de les utiliser, comme celles réécrivant les textes de tous les labels liés à l'administration de ce type de contenu :

```
register_post_type(
  'recipes',
  array(
    'label'              => _x('Recipes', 'General name for "Ad" post type'),
    'labels'             => array(
      'name'             => _x('Recipes', 'Plural name for "Recipes" post type'),
      'singular_name'    => _x('Recipe', 'Singular name for "Recipes" post type'),
      'add_new'          => __('Add New'),
      'add_new_item'     => __('Add New Recipe'),
      'edit_item'        => __('Edit Recipe'),
      'view_item'        => __('View Recipe'),
      'search_items'     => __('Search Recipes'),
      'not_found'        => __('No recipes found'),
      'not_found_in_trash'=> __('No recipes found in trash')
    ),
    'public'             => true,
    'hierarchical'       => true,
    'menu_position'      => 20,
    'supports'           => array( 'title', 'editor', 'comment' )
  );
);
```

Ici encore, nous avons écrit les différents labels en anglais puis les avons internationalisés, à l'aide de `__()` mais également de `_x()`, histoire d'introduire un peu de variété (l'apport de cette fonction est de pouvoir donner des indications aux traducteurs). Le domaine a de nouveau été omis par souci de place...

Toutes les options définies dans le tableau PHP sont facultatives, mais, les valeurs par défaut n'aidant pas à l'intégration au sein de l'administration, il nous faut être exhaustif. Voici le sens de celles que nous avons choisi de redéfinir :

- **La chaîne label et le tableau labels.** Même utilité que pour les taxinomies, à savoir faire en sorte que l'interface soit toujours adaptée au nom de notre nouveau type de contenu.

- **public.** Par défaut, le TCP ne s'affiche pas et n'est pas exploitable. Cela peut être utile en phase de test, mais dans notre cas nous voulons voir l'interface, lancer des recherches, etc. De fait, `true`.
- **hierarchical.** Le contenu peut être hiérarchisé si nous le souhaitons, pour avoir des pages et sous-pages dans l'interface. Sans cela, le contenu est "plat", et les pages ne peuvent pas être liées entre elles.
- **menu_position.** Où veut-on voir l'option du menu apparaître ? Sous Articles ? Sous Médias ? Sous Pages ? Ces options sont identifiées de cinq en cinq (Articles = 5, Médias = 10, Pages = 15...), donc il faut définir le nombre qui nous correspond. Si l'on souhaite que ce contenu soit en première position, on indique 0. S'il peut sans problème se caler en fin de menu, alors, on le laisse tel quel. Ici, nous souhaitons qu'il apparaisse sous les Pages, donc 20.
- **supports**. Par défaut, les nouveaux types ne donnent au rédacteur qu'une interface de création minimale : titre et éditeur. Vous pouvez demander à y ajouter le gestionnaire de commentaire, comme nous l'avons fait, ou encore les révisions, l'extrait, les rétroliens, etc.

Dès l'activation de l'extension ou du thème contenu dans cette ligne, l'interface de WordPress sera modifiée pour donner accès à une nouvelle option de menu pointant vers les pages, vous permettant de lister l'ensemble des recettes et d'en ajouter une nouvelle (voir Figure 11.05).

Figure 11.05
L'interface générée automatiquement par WordPress pour gérer le type de contenu.



Notez que ce nouveau type de contenu ne dispose pas de catégories ou de mots-clefs... Il vous est bien entendu parfaitement possible d'ajouter quelques taxinomies (hiérarchisées ou non) uniquement pour ce type de contenu ! Vous pouvez créer autant de types de contenu et de taxinomies que nécessaire : pourquoi en rester aux seuls bêtes mots-clefs et catégories ?

Comme d'habitude, le Codex détient toutes les informations sur les options disponibles : **http://codex.wordpress.org/Function_Reference/register_post_type**.

12 Construire une extension évoluée

Objectif de l'extension

Dans ce chapitre, nous allons créer une extension "évoluée", autrement dit une extension ajoutant plusieurs fonctionnalités et faisant appel à différentes API de WordPress.

Ici, nous allons mettre au point une extension permettant de gérer une liste de petites annonces. Cette extension, nommée Simple Classifieds (en anglais, "petite annonce" s'écrit *classified ads*), gère différentes fonctionnalités que nous détaillerons dans la section suivante.

C'est typiquement le genre d'extension qui pourra être utilisée dans le cadre d'un service en ligne gratuit, par exemple. Une orientation qui tend plus vers le CMS que l'outil de blog…

Quelles fonctionnalités ?

Avant d'écrire ce code, il est impératif que vous vous posiez quelques questions. Ces questions vont permettre de regrouper les fonctionnalités de votre extension dans différents groupes, selon le type de fonctions sur lesquelles vous travaillerez, mais elles doivent également rappeler les différentes pages de l'extension.

Voici une liste non exhaustive des questions qui peuvent être importantes :

- Quelles sont les fonctionnalités ?
- Quelles sont les pages indispensables de l'interface d'administration ?
- Quels utilisateurs de WP vont pouvoir modifier les réglages de votre extension ?
- Quels utilisateurs de WP vont pouvoir utiliser l'extension ?
- Comment s'affichera le contenu de l'extension dans le thème ?
- Quelle stratégie vis-à-vis du référencement ?
- Où stocker les données ?
- Quelles fonctionnalités seront disponibles depuis la page de rédaction ? Etc.

Cette liste est un simple exemple. Avec l'expérience que vous allez acquérir, vous ajouterez probablement de nouvelles questions au fur et à mesure de vos développements.

Revenons à notre exemple, Simple Classifieds, et répondons aux questions suivantes.

Quelles sont les fonctionnalités ?

L'extension doit pouvoir gérer l'ajout, l'édition et la suppression des annonces. L'extension contiendra également un widget pour afficher les dernières annonces et un shortcode permettant d'implémenter un formulaire d'ajout public dans la page de notre choix.

Quelles sont les pages indispensables de l'interface d'administration ?

Trois pages sont à prévoir : une page de gestion listant les événements, une page contenant le formulaire d'ajout et d'édition des événements et une page pour les réglages de l'extension. Le listage, l'ajout et l'édition des annonces seront directement gérés par WordPress.

Quels utilisateurs de WP vont pouvoir modifier les réglages de votre extension ?

Seul le rôle administrateur aura la possibilité de modifier les réglages de l'extension.

Quels utilisateurs de WP vont pouvoir utiliser l'extension ?

Les rôles administrateur et éditeur auront la possibilité d'utiliser l'extension, soit ajouter, effacer et modifier les événements.

Comment s'affichera le contenu de l'extension dans le thème ?

Pour afficher les annonces dans le thème de l'utilisateur, vous disposez de différents moyens. Tout d'abord, l'extension se chargera de modifier la Boucle de la page d'accueil pour y afficher les annonces plutôt que les articles de l'installation WordPress. Par ailleurs, vous avez un widget qui affiche les dernières annonces ajoutées.

Quelle stratégie vis-à-vis du référencement ?

L'extension doit être capable de s'adapter à la forme des liens de WordPress. Si les permaliens sont actifs, l'extension doit générer de belles adresses, sinon elle produira des adresses classiques avec des paramètres.

Où stocker les données ?

Les annonces seront stockées dans les tables standard de WordPress, grâce à l'utilisation d'un type de données personnalisé. Sans cela, il nous aurait fallu créer nos propres tables de base de données...

Quelles fonctionnalités seront disponibles depuis la page de rédaction ?

Il y a deux zones possibles de rédaction : la page interne de WordPress, et le formulaire généré par le shortcode de l'extension. Dans les deux cas, l'utilisateur ne pourra saisir qu'un titre et un contenu textuel.



L'exemple que nous utilisons se base sur les types de contenus personnalisés, ce qui simplifie grandement l'extension, tant du point de vue de l'accès à la base que de la mise en place des pages d'administration, qui sont quasi totalement pris en charge par WordPress.
Si vous souhaitez découvrir une extension qui vous explique comment manipuler la base ou mettre en place vos propres pages d'administration, le CD de ce livre contient la première édition du présent chapitre, qui comprend une extension de gestion d'événements, Simple Events. Notez que Simple Events a été écrit pour WordPress 2.7 et ne profite donc pas des nombreuses améliorations des versions suivantes...

Regroupement des fonctionnalités

Une fois que vous avez répondu à ces différentes questions, vous devez regrouper les données par entité logique de WordPress. Une entité logique correspond en fait à un groupement de fonctionnalités selon les bibliothèques de fonctions utilisées ; par exemple, si l'on devait accéder à la base de données, il faudrait passer par l'API WPDB.

Administration

- Ajout des menus dans WordPress.
- Création des nouvelles permissions et attributions aux rôles.
- Création du contenu des trois pages de l'extension.
- Mise en place d'un shortcode permettant de placer un formulaire d'ajout à une page arbitraire.

Accès en base de données

- Totalement pris en charge par WordPress grâce aux types de contenu personnalisés.

Réécriture d'adresse Internet – Rewriting

- Totalement pris en charge par WordPress.

Fonction du thème

- Modification à la volée de la Boucle de la page d'accueil.

Widget

- Création de l'option permettant d'activer ou non le widget.
- Création du widget.

Taxinomie

- Enregistrement de deux taxinomies pour notre type de contenu personnalisé : catégories et mots-clefs.

Architecture de l'extension

Fichier ou dossier ?

Pour architecturer une extension WordPress, deux cas de figure se présentent.

Le premier cas, c'est celui d'une extension simple comme Hello Dolly, que vous pouvez retrouver par défaut dans WordPress. Comme vous avez pu le constater, cette extension ne fait pas grand-chose, de ce fait placer toutes les fonctions dans un seul et même fichier est largement compréhensible.

Le second cas, à l'instar de notre exemple ici, est celui d'une extension évoluée, susceptible de contenir différents fichiers pour la traduction, un widget, une documentation ou encore des modèles de templates. Pour cette raison, il est nécessaire de créer un dossier pour ce type d'extension.

Un gros fichier ? Ou plusieurs petits fichiers ?

Dans le cadre d'une extension assez complexe, il y a deux façons de répartir le code. Soit vous placez tout le code dans un seul fichier PHP, soit vous découpez l'extension par entité logique pour créer un fichier PHP par entité. Les deux techniques ont leurs avantages et leurs défauts.

Par exemple, si vous choisissez de placer toutes les fonctions dans un seul et même fichier, il sera difficile de trouver rapidement une fonction. Une extension un tant soit peu complexe atteindra très rapidement les 1 000, voire 2 000 lignes, pour peu que vous ayez des besoins qui ne soient pas pris en charge par les API de WordPress. D'un autre côté, même avec un grand nombre de fonctions, si ces dernières sont réparties dans différents fichiers thématiques, il sera assez facile de trouver celles qui vous intéressent.

Néanmoins, il n'est pas toujours possible de découper toutes les fonctions dans différents fichiers, pour la simple et bonne raison qu'un développeur peut travailler avec des classes PHP, et ces dernières ne peuvent pas être réparties sur plusieurs fichiers. Dans ce cas précis, la répartition pourra être la suivante : une classe cliente lancée tout le temps, et une classe admin ne démarrant que les fonctions spécifiques à l'interface d'administration seulement lorsque vous vous y trouvez.

Architecture de Simple Classifieds

Ici, pour simplifier le code, l'extension sera principalement développée *via* des fonctions classiques ; seul le widget sera conçu sous la forme d'une classe. De ce fait, l'extension sera architecturée de la façon suivante :

```
simple-classifieds/
  simple-classifieds.php
  readme.txt
  languages/
    simpleclassifiedsfr_FR.mo
    simpleclassifieds-fr_FR.po
    simpleclassifieds.pot
  inc/
    classifieds.init.php
    classifieds.options.php
    classifieds.shortcodes.php
    classifieds.widgets.php
```

Toute l'extension est contenue dans le dossier simple-classifieds. Ce dernier possède deux dossiers et un fichier PHP.

Le fichier PHP simple-classifieds.php contient l'en-tête spécifique aux extensions WordPress. C'est lui qui gère l'inclusion des autres bibliothèques, autrement dit c'est le cœur de l'extension.

Les deux dossiers sont languages et inc. Ils contiennent respectivement les fichiers de traduction de l'extension et les bibliothèques de fonctions nécessaires à l'extension, qui sont au nombre de quatre.

Développement de l'extension

Les bases de l'extension

L'en-tête de WordPress

Comme dans toute extension, il est nécessaire d'avoir l'en-tête des extensions WordPress.

Nous débuterons le fichier simple-classifiedss.php avec ce code :

```
/*
Plugin Name: Simple Classifieds
Plugin URI: http://wordpress.org/extend/plugins/simple-classifieds/
Description: Run your own classifieds website using WordPress.
Version: 1.0
Author: Xavier Borderie
Author URI: http://www.wordpress-fr.net
License: GPL2
Text Domain: simpleclassifieds
*/
```

Cet en-tête peut être suivi de la licence de l'extension et d'une notice de copyright.

Inclusion des différents fichiers

Pour permettre l'inclusion des fonctions contenues dans les différents fichiers, vous devez ajouter les appels PHP des fichiers après l'en-tête de WordPress :

```
require( dirname(__FILE__) . '/inc/classifieds.init.php' );
require( dirname(__FILE__) . '/inc/classifieds.shortcodes.php' );
```

Seuls sont appelés ici les fichiers qui doivent forcément être chargés en même temps que l'extension. D'autres fichiers ne sont chargés qu'en cas de besoin...

Remarquez que nous ne précisons pas le chemin complet des fichiers PHP, nous utilisons à la place la fonction PHP `dirname()` qui retourne le chemin complet du dossier contenant le fichier que nous passons en paramètre. Ici nous passons en paramètre la constante PHP `__FILE__` ; cette constante a comme valeur le chemin complet du fichier où la constante est appelée.

Activation de l'extension

Lors de l'activation de l'extension dans WordPress, nous allons devoir effectuer deux actions : créer le type de contenu et les taxinomies utilisées par nos petites annonces, et initialiser des permissions spécifiques à notre extension.

Les fonctions créant le type de contenu et les taxinomies sont déclenchées par le crochet 'init' :

```
add_action( 'init', 'xb_classifieds_build_post_type' );
add_action( 'init', 'xb_classifieds_build_taxonomies' );
```

Ainsi, ces deux fonctions seront lancées tant que l'extension sera activée.

Pour lancer une fonction PHP uniquement lors de son activation (et donc éviter les multiples appels alors que le code est déjà en place), nous utiliserons la fonction `register_activation_hook()`, qui appellera automatiquement la fonction `xb_classifieds_build_permissions()` lors de l'activation de l'extension.

Ajoutez cette ligne :

```
register_activation_hook( __FILE__, 'xb_classifieds_build_permissions' );
```

Cette fonction doit être placée après l'ajout des fichiers PHP.

Initialisation des permissions

L'extension disposera de deux permissions différentes : une première permettant de gérer les données de l'extension et l'autre permettant de modifier les réglages de l'extension.

Ici, la permission de gestion sera nommée `use_classifieds` et elle sera attribuée aux rôles éditeur et administrateur, tandis que la permission `admin_classifieds` sera attribuée uniquement à l'administrateur, lui permettant de modifier les réglages :

```
// Ajout des permissions
function xb_classifieds_build_permissions() {
  if ( function_exists('get_role') ) {

    // Je récupère l'objet "Rôle administrateur"
    $role = get_role('administrator');

    // Si la permission "use_classifieds" n'existe pas, on l'ajoute.
    if( $role != null && !$role->has_cap('use_classifieds') ) {
      $role->add_cap('use_classifieds');
    }

    // Pareil pour la permission "admin_classifieds"
    if( $role != null && !$role->has_cap('admin_classifieds') ) {
      $role->add_cap('admin_classifieds');
    }

    // On supprime la variable de notre fonction.
    unset($role);

    // On procède de la même façon pour le rôle "Editeur" sauf qu'on lui ajoute
    uniquement la permission "user_classifieds"
    $role = get_role('editor');
    if( $role != null && !$role->has_cap('use_classifieds') ) {
      $role->add_cap('use_classifieds');
    }
```

```
      // On supprime la variable de notre fonction.
      unset($role);
    }
  }
```

Ces permissions vont permettre de sécuriser les actions de l'extension et les menus. WordPress n'affichera que les menus dont l'utilisateur possède la permission. De ce fait, un éditeur ne verra pas le menu Réglages de l'extension.

Initialisation de l'extension

Pour lancer les différentes fonctionnalités de l'extension, il faut initialiser les différents filtres et actions. Pour cela, vous allez regrouper ces appels dans une seule et unique fonction xb_classifieds_init() que vous placerez dans le fichier simple-classifieds.php.

C'est la seule fonction que contiendra ce fichier et elle sera placée à la fin du fichier.

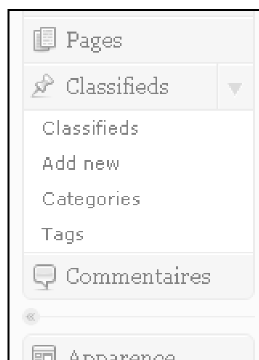
```
// Fonction d'initialisation de l'extension
add_action('plugins_loaded', 'xb_classifieds_init');
function xb_classifieds_init() {
  //…
}
```

Vous ajouterez du code à l'intérieur de la fonction au fur et à mesure du développement. Notez que la fonction sera lancée lors de l'action `plugins_loaded`, pas avant !

Mise en place du type de contenu

Nos petites annonces seront stockées non pas au sein d'articles ou de pages, mais dans un nouveau type de contenu créé pour l'occasion et nommé "classifieds" (petite annonce en anglais, voir Figure 12.01).

Figure 12.01
Le menu généré par WordPress, avec les taxinomies.



La création d'un type de contenu se fait très rapidement : il suffit d'appeler la fonction `register_post_type()`, et le type de contenu est pris en compte par WordPress, qui

génère automatiquement les menus de l'interface d'administration, ainsi que toute la logique interne nécessaire au bon fonctionnement de ce type de contenu.

Parce que nous sommes attentifs aux détails, nous allons faire en sorte d'intégrer au mieux le type de l'interface. Cela signifie non seulement reprendre tous les labels par défaut pour les remplacer par des labels adaptés à notre type, mais également internationaliser les chaînes, afin de permettre la traduction de notre extension.

Voici notre déclaration complète de type de contenu :

```
$labels = array(
  'name'               => __('Classifieds', 'simpleclassifieds'),
  'singular_name'      => __('Classified', 'simpleclassifieds'),
  'add_new'            => __('Add new', 'simpleclassifieds'),
  'add_new_item'       => __('Add New Classified', 'simpleclassifieds'),
  'edit_item'          => __('Edit Classified', 'simpleclassifieds'),
  'new_item'           => __('New Classified', 'simpleclassifieds'),
  'view_item'          => __('View Classified', 'simpleclassifieds'),
  'search_items'       => __('Search Classifieds', 'simpleclassifieds'),
  'not_found'          => __('No classifieds found', 'simpleclassifieds'),
  'not_found_in_trash' => __('No classifieds found in trash',
'simpleclassifieds')
);
$args = array(
  'label'             => __('Classifieds', 'simpleclassifieds'),
  'labels'            => $labels,
  'public'            => true,
  'menu_position'     => 20,
  'supports'          => array( 'title', 'editor', 'author' ),
  'show_in_nav_menus' => false
);
register_post_type( 'classified', $args );
```

Notre type est donc identifié sous le nom interne "classified".

Notez que nous n'avons modifié que les options dont le réglage par défaut ne nous arrangeait pas. Par exemple, par défaut un type de contenu n'est pas public, afin de pouvoir le développer sans pour autant troubler le fonctionnement de l'interface. Grâce à `menu_position`, nous plaçons le menu de gestion des petites annonces après celui de gestion des pages. L'écran d'ajout/modification d'une annonce ne contiendra que le titre, l'éditeur de texte et la sélection d'auteur. Enfin, les annonces ne s'afficheront pas dans le menu de navigation. De nombreuses autres options sont disponibles, mais leurs valeurs par défaut nous suffisent pour le moment.

Pour assurer la traduction de toutes nos chaînes, nous les avons placées au sein de la fonction `__()`, en précisant le *text-domain* de notre extension en second argument : "simpleclassifieds".

Mise en place des taxinomies

Notre type de données est personnalisé, mais nous souhaitons tout de même que celui-ci puisse profiter des catégories et mots-clefs. Nous allons devoir recréer ces taxinomies pour le type "classified", ce qui n'est guère plus difficile que créer le type de contenu lui-même.

La fonction à utiliser cette fois est `register_taxonomy()` et, tout comme `register_post_type()`, elle suffit pour mettre en place une interface et une logique complexes mais dispose de nombre d'options.

Voici le code complet mettant en place les catégories de petites annonces, et les mots-clefs de petites annonces :

```
register_taxonomy(
  'classifieds_categories',
  'classified',
  array(
    'hierarchical' => true,
    'label'              => __('Categories', 'simpleclassifieds'),
    'labels'             => array(
      'name'             => __('Categories', 'simpleclassifieds'),
      'singular_name     => __('Category', 'simpleclassifieds'),
      'search_items      => __('Search Categories', 'simpleclassifieds'),
      'popular_items     => __('Popular Categories', 'simpleclassifieds'),
      'all_items'        => __('All Categories', 'simpleclassifieds'),
      'parent_item'      => __('Parent Category', 'simpleclassifieds'),
      'parent_item_colon' => __('Parent Category:', 'simpleclassifieds'),
      'edit_item'        => __('Edit Category', 'simpleclassifieds'),
      'update_item'      => __('Update Category', 'simpleclassifieds'),
      'add_new_item'     => __('Add New Category', 'simpleclassifieds')
    ),
    'query_var'          => true,
    'rewrite'            => true
  )
);

register_taxonomy(
  'classifieds_tags',
  'classified',
  array(
    'hierarchical'       => false,
    'label'              => __('Tags', 'simpleclassifieds'),
    'labels'             => array(
      'name'             => __('Tags', 'simpleclassifieds'),
      'singular_name'    => __('Tag', 'simpleclassifieds'),
      'search_items'     => __('Search Tags', 'simpleclassifieds'),
      'popular_items'    => __('Popular Tags', 'simpleclassifieds'),
      'all_items'        => __('All Tags', 'simpleclassifieds'),
      'parent_item'      => __('Parent Tag', 'simpleclassifieds'),
      'parent_item_colon' => __('Parent Tag:', 'simpleclassifieds'),
      'edit_item'        => _(_'Edit Tag', 'simpleclassifieds'),
      'update_item'      => __('Update Tag', 'simpleclassifieds'),
      'add_new_item'     => _('Add New Tag', 'simpleclassifieds'),
      'separate_items_with_commas' => __('Separate tags with commas',
'simpleclassifieds'),
      'add_or_remove_items'        => __('Add or remove tags',
'simpleclassifieds'),
      'choose_from_most_used'      => __('Choose from must used tags',
'simpleclassifieds'),
    ),
    'query_var'          => true,
    'rewrite'            => true
  )
);
```

Ici encore, nous réécrivons l'ensemble des labels de chaque taxinomie, mais ne modifions que les options dont les valeurs par défaut ne nous conviennent pas. Seule exception, l'option hierarchical : celle-ci est par défaut false mais, pour simplifier la différenciation de chaque taxinomie à la lecture du code, nous indiquons explicitement sa valeur.

Notez par ailleurs que la taxinomie non hiérarchisée (les mots-clefs) dispose de trois labels supplémentaires à réécrire.

Comme il se doit, nos deux taxinomies sont, par leur deuxième argument, assignées au type de nos petites annonces, "classified".

Partie – Administration

Menu de WordPress

Grâce à la "magie" de la gestion par WordPress des types de contenu personnalisés et des taxinomies personnalisées, le plus gros du travail est fait côté administration : dès la mise en place de ces derniers, WordPress a généré l'ensemble des menus nécessaires, ainsi que la logique qui intègre l'ensemble à WordPress.

Il nous reste néanmoins un menu à mettre en place : celui qui permettra de gérer quelques réglages de notre extension.

Pour ajouter ce menu dans WordPress, vous allez lier une fonction à l'action `admin_init`, qui définira nos options et les sections où elles s'affichent, et une autre fonction à l'action `admin_menu`, qui insérera le menu.

Donc, dans un premier temps, il nous faut ajouter ces actions dans la fonction d'initialisation xb_classifieds_init(), mais qui seront lancées uniquement dans l'interface d'administration :

```
// Initialisation Admin
if ( is_admin() ) {
  require( dirname(__FILE__) . '/inc/classifieds.options.php' );
  add_action( 'admin_init', 'xb_classifieds_options_init' );
  add_action('admin_menu', 'xb_classifieds_options_add_page');
}
```

Nous verrons le contenu de la fonction `xb_classifieds_options_init()` dans la section suivante. Voyons tout d'abord celui de la fonction `xb_classifieds_options_add_page()`, qui ajoute le menu :

```
function xb_classifieds_options_add_page() {
  add_options_page(
    __('Classifieds'),                  // Titre de la page
    __('Classifieds'),                  // Titre du menu de la page
    'admin_classifieds',                // Capacité nécessaire pour y accéder
    'classifieds_options',              // Identifiant/URL de la page
    'xb_classifieds_options_build_page' // Fonction construisant la page
  );
}
```

Cette fonction est placée dans le fichier classifieds.options.php.

La fonction appelée nécessite la permission `admin_classifieds`, que seul le rôle administrateur possède.

Réglages – Options des petite annonces

Dans cette extension, la page des options se veut assez sommaire (voir Figure 12.02). En effet, elle ne présente que deux cases à cocher : l'affichage ou non des petites annonces sur la page d'accueil du site, l'activation ou non du widget. Les réglages seront stockés sur la table des options de WordPress, grâce à l'API des options (Settings API).

Figure 12.02

Rendu de la page de réglages de l'extension.

Enregistrer les options dans la base de données

Occupons-nous de notre fonction `xb_classifieds_options_init()`, exploitant la Settings API de WordPress, qui simplifie grandement la mise en place des formulaires d'options. Pour cela, nous allons commencer par enregistrer les réglages de notre extension :

```
register_setting(
  'classifieds_options',
  'classifieds_options',
  'xb_classifieds_options_sanitize'
);
```

L'enregistrement requiert au moins deux arguments : le groupe d'options que nous définissons, la variable de l'option, et la fonction qui validera le contenu de l'option. Dans notre cas, notre variable sera un tableau PHP, ce qui nous permet de stocker de nombreuses valeurs sans pour autant trop remplir la base de données. Nous verrons plus loin le contenu de la fonction de validation des données...

Définir le contenu de la page d'options

À l'aide des fonctions de la Settings API, nous allons définir totalement le contenu de notre page, qui sera ensuite générée par WordPress, ce dernier prenant en charge les aspects les plus laborieux du formulaire.

Voici donc comment nous mettons en place deux sections pour notre page, avec une option dans chaque section :

```
add_settings_section(
  'classifieds_general_settings',
  __('General Settings', 'simpleclassifieds'),
  'xb_classified_general_settings_text',
  'classifieds_section'
);
add_settings_field(
  'classified_adsInHome_setting',
  __('Display the ads on the homepage?', 'simpleclassifieds'),
  'xb_classifieds_displayAdsInHome_setting',
  'classifieds_section',
  'classifieds_general_settings'
);

add_settings_section(
  'classifieds_widgets_settings',
  __('Widget Settings', 'simpleclassifieds'),
  'xb_classified_widets_settings_text',
  'classifieds_section'
);
add_settings_field(
  'classified_latestWidget_setting',
  __('Activate "Latest Classifieds" widget?', 'simpleclassifieds'),
  'xb_classifieds_displaylatestWidget_setting',
  'classifieds_section',
  'classifieds_widgets_settings'
);
```

La fonction de mise en place d'une section prend quatre arguments : un identifiant unique, le titre de la section (qui sera affiché dans la page), une fonction qui permettra d'afficher du contenu pour la section, et enfin le nom de la page à laquelle s'applique cette fonction.

De son côté, la fonction de mise en place d'une option (d'un "champ") prend cinq arguments : un identifiant unique (qui sera également appliqué aux balises HTML de cette option), le label de l'option, une fonction qui affichera le code HTML de notre balise INPUT, et enfin l'identifiant de la section de page dans laquelle cette option sera affichée.

Comme vous le constatez, nous plaçons l'option d'affichage des annonces en page d'accueil dans une section "General", et l'option d'activation du widget dans une autre section, "Widget".

Remplir la page d'options

Pour notre exemple, les fonctions affichant du contenu pour les sections ne feront que sortir un court descriptif. Mais rien n'empêche de rajouter d'autres choses : images, liens, etc. Voici comment nous nous y prenons :

```
function xb_classified_widgets_settings_text() {
  echo '<p>' . __('These settings apply to the extension\'s widgets.',
'simpleclassifieds') . '</p>';
}
function xb_classified_general_settings_text() {
  echo '<p>' . __('These settings apply to the extension in general.',
'simpleclassifieds') . '</p>';
}
```

Les options de notre formulaire, elles, doivent explicitement être codées :

```
function xb_classifieds_displaylatestWidget_setting() {
  $options = get_option('classifieds_options');
  if ( !isset($options['latest_widget']) ) {
    $options['latest_widget'] = 0;
  }
  ?>
  <input id='classified_latestWidget_setting' name="classifieds_options[latest_
widget]" type="checkbox" value="1" <?php checked('1', $options['latest_
widget']); ?> />
  <?php
}

function xb_classifieds_displayAdsInHome_setting() {
  $options = get_option('classifieds_options');
  if ( !isset($options['ads_in_home']) ) {
    $options['ads_in_home'] = 0;
  }
  ?>
  <input id='classified_adsInHome_setting' name="classifieds_options[ads_in_
home]" type="checkbox" value="1" <?php checked('1', $options['ads_in_home']);
?> />
  <?php
}
```

Ces deux fonctions seront appelées lors de la génération de la page entière, que nous allons bientôt voir. Chacune est définie dans la fonction `add_settings_field()` de l'option, et elles suivent le même cheminement :

- Récupération du tableau des réglages de notre extension, nommé "classifieds_options" (comme nous l'avons déjà vu).
- Vérification de l'existence d'une valeur pour notre option. S'il n'y en a pas, nous l'initialisons.
- Affichage du code HTML du champ (dans notre cas, une case à cocher).

Notez que l'attribut ID de la balise INPUT doit correspondre à l'identifiant du champ de l'option, tel que défini en premier argument de la fonction `add_settings_field()`.

Par ailleurs, afin de récupérer un tableau de valeurs plutôt que deux valeurs séparées, l'attribut NAME de la balise prend une forme particulière : `nom_du_tableau_d'options[nom_de_l'option]`.

Enfin, nous faisons usage de la fonction `checked()` de WordPress, qui prend en charge la logique d'affichage de l'attribut CHECKED ou non, selon que le premier argument est égal au second.

Vérifier les valeurs données par l'utilisateur

Tels qu'ils sont conçus, nos deux champs renvoient une chaîne contenant la valeur 1 ou 0. Pour notre exemple, nous préférons un entier plutôt qu'une chaîne. Il nous faut donc convertir la valeur avant l'enregistrement dans la base de données. C'est ici qu'entre en jeu la fonction de nettoyage/validation, que nous avons définie en quatrième argument de la fonction `register_setting()` ! Voici le code que nous lui donnons :

```
function xb_classifieds_options_sanitize($input) {
  // $input est un tableau contenant toutes nos options.
  // Il nous faut appliquer une vérification à toutes les options qui en ont
besoin.
  if ( isset($input['latest_widget']) ) {
    $input['latest_widget'] = ( $input['latest_widget'] == 1 ? 1 : 0 );
  }
  if ( isset($input['ads_in_home']) ) {
    $input['ads_in_home'] = ( $input['ads_in_home'] == 1 ? 1 : 0 );
  }
  return $input;
}
```

La même conversion est appliquée aux deux valeurs, mais tout peut se faire. Par exemple, dans le cadre d'un champ texte où l'on demande une adresse e-mail, il est possible d'utiliser une expression régulière afin de vérifier qu'il s'agit bien d'une adresse valide.

Afficher la page d'options

Toute cette préparation aboutit enfin à la génération du code de notre page d'options, notamment son formulaire. Cela peut sembler laborieux, mais le gain de temps est certain : là où auparavant le développeur devait gérer à la main les annonces et le contenu de la variable `$_POST` pour traiter correctement le formulaire, désormais WordPress prend en charge toutes ces étapes, laissant le développeur se concentrer sur son code.

Voici le code de génération du formulaire de la page, appelé par `add_options_page()` :

```
function xb_classifieds_options_build_page() {
  ?>
  <div class="wrap">
    <div class="icon32" id="icon-options-general"><br /></div>
    <h2><?php _e( 'Classifieds Options', 'simpleclassifieds' );?></h2>
    <?php _e( 'Options for the Simple Classifieds extension.', 'simple-
    classifieds' );?>
    <form method="post" action="options.php">
      <?php
      settings_fields('classifieds_options');
      do_settings_sections('classifieds_section');
      ?>
      <p class="submit">
        <input type="submit" class="button-primary" value="<?php _e('Update
        Settings') ?>" />
      </p>
```

```
    </form>
  </div>
<?php
}
```

Certes, il reste des balises FORM et quelques attributs à mettre en place. Malgré cela, toutes les étapes de préparation aboutissent aux deux fonctions au cœur du formulaire :

- settings_field() se charge de mettre en place le code HTML des annonces et autres champs nécessaires au bon fonctionnement du formulaire.
- do_settings_sections() affiche les différentes sections et leurs champs, dans l'ordre que nous avons défini, et ici encore avec les bonnes valeurs mises en place.

Le gain de temps est assez phénoménal, et dans le cadre de page d'options à rallonge il est d'autant plus appréciable de pouvoir déléguer la prise en charge des tâches laborieuses au CMS, plutôt que tout gérer à la main. Les options sont sauvegardées et récupérées, les valeurs des balises sont mises à jour, WordPress vous notifie cette mise à jour... La Settings API est extrêmement puissante.

Exploiter les options

Une fois la page d'options mise en place, il reste à appliquer les choix de l'utilisateur dans le code. Dans notre exemple, nous le faisons directement depuis la fonction xb_classifieds_init(). Voici la procédure :

```
$options = get_option( 'classifieds_options' );
if (  isset($options['latest_widget']) && $options['latest_widget'] === 1 ) {
  add_action( 'widgets_init', create_function('', 'return register_
widget("Classifieds_Latest_Widget");') );
}

if ( isset($options['ads_in_home']) && $options['ads_in_home'] === 1 ) {
  add_action( 'pre_get_posts', 'change_home_loop' );
}
```

Dans les deux cas, nous vérifions que la valeur cherchée est bien définie, puis qu'elle correspond à ce que l'on attend. Les fonctions voulues sont alors déclenchées, ici par une action pour chaque option : la première enregistre le widget de notre extension, la seconde modifie la Boucle de la page d'accueil, grâce à la fonction change_home_loop(), que voici :

```
function change_home_loop( $current_query ) {
  if ( $current_query->is_home() ) {
    $current_query->query_posts( array( 'post_type' => 'classified' ) );
  }
}
```

Nous utilisons les actions les plus proches de nos besoins : widgets_init se déclenche lors de la mise en place des widgets, et pre_get_posts se déclenche lors de l'affichage de l'en-tête du thème... À noter également que pre_get_posts a l'avantage de passer en référence la requête en cours, que l'on peut donc exploiter à loisir et proprement.

Partie – Widget

Lancement du widget

Par défaut, le fichier PHP du widget classifieds.widget.php n'est pas chargé par l'extension.

Les fonctions propres aux widgets doivent être chargées lors de l'événement `widgets_init`. Vous chargerez le fichier PHP depuis la fonction `xb_classifieds__init()` uniquement si l'option de l'extension est activée.

Ajoutez ceci dans la fonction `event_init()` :

```
$options = get_option( 'classifieds_options' );
if (  isset($options['latest_widget']) && $options['latest_widget'] === 1 ) {
  include( dirname(__FILE__) . '/inc/classifieds.widgets.php' );
  add_action( 'widgets_init', create_function('', 'return register_
widget("Classifieds_Latest_Widget");') );
}
```

Notre action doit appeler une fonction mais, vu que nous voulons simplement enregistrer notre widget, nous passons par une fonction lambda (une fonction sans nom/anonyme), créée pour l'occasion par `create_function()`. Celle-ci se charge d'appeler à son tour `register_widget()`. Il n'est pas obligatoire de passer par une fonction lambda, mais cela nous évite simplement de devoir créer plus de code que nécessaire...

Création du widget

Un widget dans WordPress se décompose en deux parties : une partie cliente qui affiche le widget sur le thème et une partie admin qui gère les options et l'affichage dans l'interface d'administration. Le code des widgets est exclusivement situé dans le fichier classifieds.widgets.php (voir Figure 12.03).

Figure 12.03
Rendu du widget côté thème.



L'API de widgets de WordPress nécessite de passer par une classe PHP qui étend l'objet `WP_Widget` de WordPress. Cette classe soit contenir quatre méthodes standard : la méthode de construction (du même nom que la classe), puis `form()` pour le formulaire d'option, `update()` pour les actions à lancer lors de la mise à jour du widget, et `widget()` pour le contenu du widget lui-même.

Méthode de construction

L'instanciation de notre classe déclenche automatiquement la fonction éponyme de la classe :

```
class Classifieds_Latest_widget extends WP_Widget {
  function Classifieds_Latest_widget() {
    $this->WP_Widget(
      'classifieds',
      __('Latest classifieds', 'simpleclassifieds'),
      array(
        'classname' => 'widget-classifieds',
        'description' => __( 'Displays the latest classifieds from Simple
         Classifieds', 'simpleclassifieds' )
      )
    );
  }

  // suite du code
}
```

Celle-ci se charge de préciser certains attributs de notre widget : un identifiant unique, le nom qui sera affiché dans la page de configuration, et un premier tableau d'arguments contenant le nom de l'attribut CLASS appliqué au widget, et sa description telle qu'affichée sur la page de configuration. Il est possible d'ajouter en quatrième argument un second tableau contenant la largeur en pixels (`'width'`, nécessaire si elle doit dépasser les 250 pixels) et la hauteur (`'height'`, inutilisé à l'heure actuelle).

Les champs du widget

Nous allons dès à présent définir les données qui pourront être modifiées par le widget au sein d'une méthode à part afin de les rendre accessibles aux différentes autres méthodes de la classe. Elles seront contenues par un tableau associatif.

Nous souhaitons que l'utilisateur puisse modifier deux aspects du widget : le nombre d'annonces affichées, et le titre utilisé dans la barre latérale. Pour chaque variable, nous offrons une valeur par défaut.

```
function getFields() {
  return array(
    'title' => __( 'Latest classifieds', 'simpleclassifieds'),
    'max'   => 10,
  );
}
```

Administration du widget

Le widget que manipulera l'administrateur (voir Figure 12.05) du site n'est guère plus qu'un formulaire : il reste parfaitement classique dans son utilisation de la logique PHP et des balises HTML :

```
function form( $instance ) {
  $instance = wp_parse_args( (array) $instance, $this->getFields() );
  foreach ( (array) $this->getFields() as $field => $field_value ) {
    ${$field} = esc_attr( $instance[$field] );
  }

  ?>
  <p><?php _e('Empty field will use default value.', 'simpleclassifieds'); ?></p>

  <p>
    <label for="<?php echo $this->get_field_id('title'); ?>">
      <?php _e('Title:', 'simpleclassifieds'); ?>
      <input class="widefat" type="text" id="<?php echo $this->get_field_
id('title'); ?>" name="<?php echo $this->get_field_name('title'); ?>"
value="<?php echo $title; ?>" />
    </label>
  </p>

  <p>
    <label for="<?php echo $this->get_field_id('max'); ?>">
      <?php _e('Number of classifieds:', 'simpletags'); ?>
      <select id="<?php echo $this->get_field_id('max'); ?>" name="<?php echo
      $this->get_field_name('max'); ?>">
        <option <?php selected($max, 2); ?> value="2">2</option>
        <option <?php selected($max, 4); ?> value="4">4</option>
        <option <?php selected($max, 6); ?> value="6">6</option>
        <option <?php selected($max, 8); ?> value="8">8</option>
        <option <?php selected($max, 10); ?> value="10">10</option>
      </select>
    </label>
  </p>
<?php
}
```

La méthode `form()` récupère automatiquement une instance de l'objet en premier argument. En utilisant la fonction `wp_parse_args()` de WordPress, nous pouvons remplacer les valeurs par défaut avec celles définies dans l'instance.

Ce faisant, nous créons une boucle créant à la volée une variable pour chaque variable manipulable et lui assignant la dernière valeur. Ceci fait, nous pouvons construire le formulaire lui-même.

Nous avons donc deux options à mettre en place. Chacune est encadrée d'une balise LABEL afin de lier le titre (traduisible) aux balises d'action : dans un cas, un champ textuel, dans l'autre, un sélecteur.

Dans chaque cas, nous utilisons la fonction `get_field_id()` de WordPress, permettant d'obtenir un ID propre et conforme pour la balise à partir du nom initial de l'option, et `get_field_name()` pour obtenir la même chose pour le champ NAME. Notez que les variables de nos options reprennent le nom de celles-ci, grâce à la boucle `foreach()` au début de la méthode.

Le champ texte reprend donc la valeur courante du titre.

Pour la sélection d'option, rien de bien nouveau pour qui sait coder en HTML et PHP, mis à part l'utilisation de la fonction `selected()`, qui renvoie l'attribut `SELECTED` dans le cas où ses deux arguments sont égaux.

Mise à jour des variables

Le formulaire du widget est en place, et WordPress se charge de le placer dans des balises `FORM`, avec un bouton Enregistrer, comme pour les autres widgets. Au clic sur ce bouton Enregistrer, WordPress va déclencher la méthode `update()` de la classe de ce widget. La voici, dans toute sa mirifique splendeur :

```
function update($new_instance, $old_instance) {
  $instance = $old_instance;

  foreach ( (array) $this->getFields() as $field => $field_value ) {
    $instance[$field] = strip_tags($new_instance[$field]);
  }

  return $instance;
}
```

WordPress donne à cette méthode deux valeurs pour ses arguments : le premier reçoit la nouvelle instance des variables telles que choisies par l'utilisateur, la seconde conserve l'ancienne instance de ces variables, avec les valeurs initiales. Cela nous permet de manipuler l'ensemble de manière précise, en versant le contenu de l'un dans l'autre, avec potentiellement une vérification des données si besoin est.

Une fois tous les champs correctement transvasés d'une instance à l'autre, nous renvoyons l'instance finale.

Partie publique du widget

Voyons voir maintenant l'aspect que prendra notre widget une fois dans la barre latérale. Tout ce que nous souhaitons, c'est que le widget affiche les X dernières annonces.

Lorsque le thème affiche les widgets, il appelle leur méthode `widget()`. Voici la nôtre :

```
function widget( $args, $instance ) {
  extract( $args );

  foreach ( (array) $this->getFields() as $field => $field_value ) {
    ${$field} = trim( $instance[$field] );
  }

  echo $before_widget;
  echo $before_title . apply_filters( 'widget_title', $title ) . $after_title;

  $classifieds_query = new WP_Query( array( 'post_type' => 'classified',
  'posts_per_page' => $max ) );
  echo '<ul>';
  if ( $classifieds_query->have_posts() ) :
    while ( $classifieds_query->have_posts() ) : $classifieds_query->the_post();
      echo '<li><a href="' . get_permalink() .'">' . get_the_title() . '</a></li>';
```

```
      endwhile;
    else:
      echo '<li>' . __('There is no spoon.', 'simpleclassifieds') . '</li>';
    endif;
    echo '</ul>';
    wp_reset_query();

    echo $after_widget;
  }
```

Les deux arguments de la méthode sont ici encore pris en charge par WordPress : `$args` contient différentes variables définies par la barre latérale, comme `$before_widget`, qui doit ouvrir tout widget afin de mieux s'intégrer et que nous récupérons grâce à la fonction PHP `extract()` ; et `$instance` contient notre instance des réglages.

Ici encore, une boucle foreach nous permet de récupérer des variables exploitables à part de notre instance de réglages.

L'idée principale est donc d'afficher la liste des X dernières annonces. Pour ce faire, nous instancions l'objet `WP_Query` avec deux modifications par rapport à une requête normale : on ne cible que le type "classified", et on limite le nombre d'annonces *via* la variable `$max`.

Ceci fait, nous pouvons afficher la liste des titres d'annonces comme pour n'importe quelle Boucle – sans oublier de prendre en compte le cas où il n'y aura pas d'annonce... La Boucle terminée, nous remettons à zéro la requête afin de ne pas marcher sur les plates-bandes d'une autre...

Figure 12.04
Rendu du widget côté administration.

Partie – Shortcode

Nous souhaitons donc proposer à l'administrateur du site un shortcode afin qu'il puisse facilement créer une page contenant un formulaire simple d'ajout d'annonce, pour peu qu'il soit connecté... Ce shortcode prendrait la forme suivante : `[classified_submit_form]`. À l'affichage de la page, WordPress comprendra automatiquement qu'il faut remplacer ce shortcode par du contenu plus complexe... pour peu qu'on lui indique comment faire.

Notre shortcode est entièrement stocké dans le fichier `classifieds.shortcodes.php` et se déclenche dès le chargement de ce dernier grâce à la méthode `add_shortcode()` :

```
add_shortcode('classified_submit_form', 'classified_submit_form');
```

Le premier argument définit le shortcode lui-même, tandis que le second pointe vers la fonction qui produira le contenu généré par le shortcode. Que voici, justement :

```
function classified_submit_form() {
  $form = '';

  if ( is_user_logged_in() ) {
    $form .= '<form id="new_post" name="new_post" method="post" action="'
    . site_url() . '/">';
    $form .= '  <label for="post_title">' . __( 'Classified\'s title',
    'simpleclassifieds' ) . '</label>';
    $form .= '  <input name="post_title" type="text" velue="' . __('The title
    for your classified', 'simpleclassifieds') . '" />';
    $form .= '  <br />';
    $form .= '  <label for="post_text">' . __( 'Classified\'s main text',
    'simpleclassifieds' ) . '</label>';
    $form .= '  <textarea name="post_text" style="width:100%;height:7em">
    </textarea>';
    $form .= '  <input id="submit" type="submit" value="' . __( 'Publish your
    classified', 'simpleclassifieds' ) . '" />';
    $form .= '  <input type="hidden" name="action" value="post" />';
    $form .= wp_nonce_field( 'new-post' );
    $form .= '</form>';
  } else {
    $form = '<p class="xb-error">' . __('Only logged-in users can post new
    classified ads.', 'simpleclassifieds') . '</p>';
  }

return $form;

}
```

Somme toute, un formulaire tout ce qu'il y a de plus classique. Nous vérifions avant toute chose que l'utilisateur est effectivement connecté, puis nous créons les champs nécessaires : titre et contenu, soit balise `INPUT` et balise `TEXTAREA`. Il ne faut pas oublier de générer le champ nonce avec `wp_nonce_field()`, sans quoi le formulaire ne sera pas pris en compte par WordPress.

Parfait, notre page contient un formulaire, mais maintenant il faut que WordPress traite les données envoyées *via* celui-ci. Pour ce faire, nous mettons en place une fonction qui sera déclenchée par l'action "template_redirect". Cette action se déclenche juste à la fin du fichier modèle en cours. Elle est généralement utilisée pour mettre en place une redirection...

```
add_action( 'template_redirect', 'classified_submit_code' );
```

L'action appelle une fonction qui prendra en charge le traitement des données envoyées *via* le formulaire de notre shortcode :

```
function classified_submit_code() {
  if( 'POST' != $_SERVER['REQUEST_METHOD'] || empty( $_POST['action'] )
  || $_POST['action'] != 'new_post' ) {
    if ( is_user_logged_in() ) {
      if ( isset($_POST['post_title']) && isset($_POST['post_text']) ) {
        $user        = wp_get_current_user();
        $user_id     = $user->ID;
        $title       = stripslashes($_POST['post_title']);
        $post_content  = stripslashes($_POST['post_text']);

        if ( empty( $title ) || __('The title for your classified',
        'simpleclassifieds') == $title )
          // For empty or placeholder text, create a nice title based on
          content
          $post_title = title_from_content( $post_content );
        else
          $post_title = $title;

        $the_post_array = array(
          'post_author'  => $user_id,
          'post_title'    => $post_title,
          'post_content'  => $post_content,
          'post_type'    => 'classified',
          'post_status'  => 'publish'
        );

        $post_id = wp_insert_post( $the_post_array );
        if ( $post_id != 0 && is_wp_error( $post_id ) ) {
          // Post correctly inserted !
          // Optionnaly make a redirection on success page !
        } else {
          // Post not inserted, check error
          // Optionnaly make a redirection on error page !
          // var_dump($post_id);
        }
      }
    }
  }
}
```

Comme il se doit quand il s'agit de traiter les données d'un formulaire, il faut commencer par vérifier si la page a bien été appelée par la méthode `POST` de HTTP, et si les actions de notre formulaire sont bien en place.

Ceci fait, et après avoir vérifié que l'utilisateur est bien connecté, vient le moment de récupérer les données du formulaire afin de les introduire dans la base de données. Un contenu WordPress, de n'importe quel type, nécessite trois informations pour être valide : un identifiant d'auteur, un titre et un contenu, le reste étant géré par défaut par WordPress.

Pour récupérer l'identifiant de l'utilisateur courant, il suffit de faire appel à la fonction `wp_get_current_user()`, puis d'appeler la variable ID de l'objet récupéré.

Le titre et le contenu, quant à eux, sont directement fournis par le formulaire. Dans le cas où l'auteur aurait omis le titre, nous en créons un à partir des premiers mots du contenu, à l'aide d'une fonction `title_from_content()` que nous décrirons plus loin.

Nous avons toutes nos données, il reste à créer le tableau PHP nécessaire à l'insertion d'un nouveau contenu dans la base de WordPress. Nous précisons dans ce tableau le type du contenu (`'classified'`) et son état (`'published'`).

Enfin, nous pouvons insérer le contenu dans la base, à l'aide de la fonction `wp_insert_post()`, qui prend la suite en charge. Celle-ci renvoie l'ID du nouveau contenu, ce qui nous permet de réaliser quelques vérifications finales, voire de rediriger le navigateur vers une page en cas de succès ou d'échec, avec par exemple la fonction PHP `header()`.

Reste une petite fonction à expliquer, celle qui crée un titre à partir du contenu. Notre fonction est reprise de celle utilisée dans le thème open-source P2 (**http://p2theme.com/**) :

```
function title_from_content( $content ) {
  static $strlen =  null;
  if ( !$strlen ) {
    $strlen = function_exists('mb_strlen') ? 'mb_strlen' : 'strlen';
  }
  $max_len = 40;
  $title = $strlen( $content ) > $max_len? wp_html_excerpt( $content,
  $max_len ) . '...' : $content;
  $title = trim( strip_tags( $title ) );
  $title = str_replace( "\n", " ", $title );
  return $title;
}
```

Simplement, les quarante premiers caractères du contenu sont repris pour en faire un titre, terminé par un point de suspension au besoin. Le tout utilise la fonction `wp_html_excerpt()`, créée pour ce type d'occasion.

Partie – Internationalisation

Une extension qui n'est pas internationalisée ne peut pas avoir de succès ; l'internationalisation est indispensable de nos jours. Deux méthodes sont disponibles pour charger la traduction de votre extension. Ces méthodes sont à placer dans la fonction `xb_classifieds_init()` de votre extension.

La première méthode est complètement générique, bien qu'un peu complexe :

```
$locale = get_locale(); // fr_FR
if ( !empty( $locale ) ) {
  $mofile = dirname(__FILE__) . '/languages/simpleclassifieds-'.$locale.'.mo';
  // simpleclassifieds-fr_FR.mo
  load_textdomain('simpleclassifieds', $mofile);
}
```

Ici, vous récupérez vous-même le code de la langue utilisée et vous chargez le fichier MO en précisant son chemin absolu. L'intérêt de cette méthode, c'est une parfaite compatibilité quels que soient l'emplacement de l'extension et la version de WordPress employée.

L'autre méthode, plus simple mais moins souple et implémentée plus récemment dans WordPress, consiste à utiliser la fonction `load_plugin_textdomain()`.

Vous obtenez alors :

```
load_plugin_textdomain('simpleclassifieds', false, 'simple-classifieds/
languages');
```

Il vous reste à utiliser un outil comme poEdit (http://www.poedit.net/) pour extraire les chaînes marquées avec les fonctions `__()` et `_e()` afin de les cataloguer au sein d'un fichier POT que vous stockerez dans le dossier /languages de votre extension. Les traducteurs n'auront plus qu'à enregistrer leur fichier binaire traduit (MO) sur le modèle `simpleclassifieds-fr_FR.mo` pour obtenir une extension traduite dans la langue de l'utilisateur.

Conclusion

Débuter le développement d'extensions sous WordPress est quelque chose de relativement accessible. Ici, il n'est que très peu question de développement orienté objet, il s'agit principalement de fonctions PHP couplées à un mécanisme d'actions et de filtres.

Bien entendu, il est nécessaire de bien connaître le PHP pour parvenir à vos fins… Il n'empêche qu'avec la base d'extensions existantes vous pouvez facilement, même en tant que développeur occasionnel, modifier une extension pour vos besoins. Et c'est au fur et à mesure que vous modifierez des extensions et que vous vous familiariserez avec WordPress que vous serez capable de vous lancer dans le développement d'une extension de A à Z !

Partie IV

LE CAMPUS

WordPress en mode multisite

13 Le mode multisite de WordPress

Présentation

Les utilisateurs avancés (ou intrépides) de WordPress peuvent configurer leur installation pour passer en mode multisite. Cette possibilité n'est arrivée que très récemment, avec WordPress 3.0. Auparavant existaient deux éditions de WordPress : l'édition normale, utilisable par tous pour gérer un blog, et l'édition MU, pour ceux souhaitant gérer plusieurs sites WordPress à partir d'une même installation, au lieu d'installer (et de mettre à jour) plusieurs installations distinctes.

WordPress MU, ou WordPressµ, était donc la version multi-utilisateur de la fameuse application de blogging WordPress. Aujourd'hui totalement intégrée à WordPress, cette déclinaison spécifique n'a plus lieu d'être : on ne parle donc plus de WordPress MU, mais juste de WordPress, en précisant que le logiciel est en mode multisite le cas échéant.

Ce mode multisite est l'outil idéal pour les personnes souhaitant mettre en place un grand réseau de blogs : il permet à chaque utilisateur inscrit de créer son propre site. Chacun son propre blog, cela signifie chacun son propre thème, chacun ses extensions, chacun ses utilisateurs, avec les mêmes rôles que WordPress !

Concrètement, il est possible de créer depuis une installation de WordPress autant de sites WordPress que l'on souhaite (voir Figure 13.01). Ainsi, on retrouve le logiciel WordPress en mode multisite derrière des services de blogs tout compris, tels que **WordPress.com** ou **LeMonde.fr**.

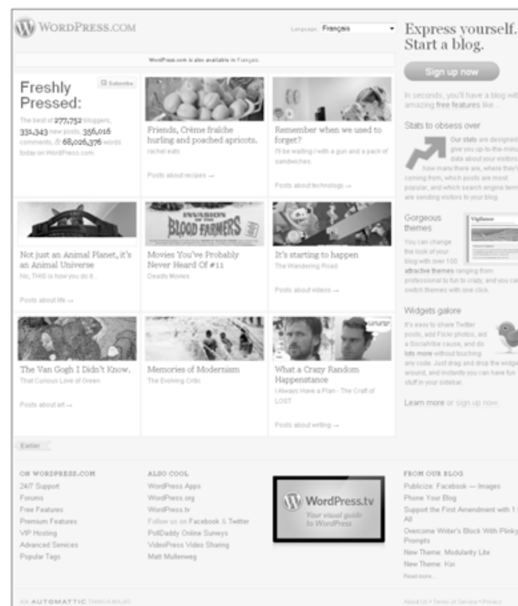


Figure 13.01
WordPress.com, une des plates-formes de blogs les plus populaires.

Historique et numérotation des versions

WordPress MU a été lancé le 4 octobre 2004 par Donncha O'Caoimh. La version 0.1 était à l'époque un fork pur et dur du logiciel WordPress ; il gérait deux fonctionnalités en plus de WordPress, dont le multiblog et l'utilisation du moteur de template Smarty pour le rendu des thèmes.

Alors que WordPress MU continuait à évoluer, Donncha a intégré l'équipe de développement d'Automattic, et WordPress MU s'est fortement rapproché de WordPress et a calqué des fonctionnalités du logiciel. Ce qui a provoqué la disparition du moteur de template Smarty.

Après plusieurs versions bêta, la première version stable de WordPress MU a été publiée le 23 octobre 2006.

Depuis cette première version 1.0, WordPress MU ne cesse d'évoluer au fil des mises à jour de WordPress. En effet, les deux logiciels partagent un code très semblable, la sortie de WPMU est généralement décalée d'une à trois semaines après celle de WordPress. WordPress MU profite ainsi de toutes les fonctionnalités de WordPress.

Jusqu'à la version 2.6 de WordPress, WordPress MU était numéroté de la même façon, à la différence que le premier chiffre était un 1 et non un 2. Par exemple, l'équivalent de WordPress 2.5 était WordPress MU 1.5. Depuis la version 2.6, WordPress MU reprend la même numérotation, soit WordPress MU 2.6.

Ce changement a été décidé pour améliorer la lisibilité de WordPress MU vis-à-vis de WordPress. Beaucoup de personnes pensaient en effet qu'il était à la traîne par rapport à WordPress. Il n'en est rien, car WordPress MU profite de toutes les innovations de WordPress.

C'est devenu d'autant plus vrai à partir de WordPress 3.0. Dès mai 2009 et la version 2.8 de WordPress, Matt Mullenweg (leader du projet WordPress) annonçait que WordPress et WPMU allaient fusionner et indiquait que cela arriverait probablement pour la version 3.0. Depuis juin 2010, c'est désormais chose faite : une petite modification de la configuration de WordPress, et l'installation peut gérer un nombre illimité de sites. Bien sûr, cela n'est recommandé que pour ceux qui maîtrisent parfaitement les tenants et aboutissants d'un tel projet...

Quels usages ?

On recense principalement deux usages de WordPress en mode multisite : plate-forme de sites publique, réseaux thématiques de sites et gros blogueurs.

Plate-forme de sites publique

Que l'inscription à la plate-forme soit payante, gratuite, ou sur invitation, WordPress permet la création de plate-forme de sites publique.

Les plates-formes les plus connues en France sont **WordPress.com**, **LeMonde.fr** et **unblog.fr**. Elles offrent à leurs visiteurs la possibilité de créer un blog pour s'exprimer sur le sujet de leur choix.

Chaque plate-forme tente de développer des fonctionnalités exclusives pour attirer les visiteurs. L'objectif d'une telle plate-forme est multiple : visibilité sur Internet, notoriété, génération de revenus *via* des publicités ou des abonnements, etc.

Une petite précision toutefois : sachez que le marché des plates-formes de blogs est extrêmement concurrentiel. Lancer une plate-forme de blogs sans innovations majeures ou sans appuis d'une communauté est voué à l'échec.

Réseaux thématiques de sites et gros blogueurs

Le second usage récurrent est l'utilisation de WordPress dans un cadre de réseaux de blogs ou de blogueurs à plusieurs blogs.

Par exemple, si un groupe scientifique international se lance dans le monde du blog, il aura tout intérêt à utiliser WordPress en mode multisite, pour permettre à chaque antenne de disposer de son propre blog.

Il suffit ensuite de créer un portail principal qui recense l'activité de tous les blogs, et on obtient d'une part un portail très dynamique, et on allège d'autre part l'administration et la maintenance des blogs.

Cette problématique fonctionne également pour les gros blogueurs. En effet, si vous disposez par exemple de trois blogs WordPress, à chaque mise à jour de WordPress ou d'une extension, il est nécessaire de dupliquer l'action trois fois. Avec WordPress en mode multisite, une fois suffira pour mettre à jour l'intégralité de vos blogs.

À qui s'adresse le mode multisite de WordPress ?

Même après fusion avec WP.org, le mode multisite de WordPress est et reste destiné à un public de connaisseurs, voire de professionnels. Si vous ne vous sentez pas à l'aise avec WordPress en mode monosite, n'imaginez même pas travailler avec le mode multisite !

Concrètement, pour utiliser WordPress en mode multisite, il faut :

- savoir manier un éditeur de texte ;
- savoir se servir d'un client FTP ;
- savoir modifier une extension, donc connaître des rudiments de PHP et de programmation WordPress ;
- parler la langue de Shakespeare, les principales ressources étant en anglais ;
- connaître les bases d'un serveur web Linux, SSH, DNS, HTTP ;
- être au fait des problèmes liés à la "scalabilité" et à l'évolution des besoins et ressources (espace serveur, bande passante, processeur, serveur dédié, etc.) ;
- très bien connaître WordPress.

Si ces connaissances vous manquent ou sont incomplètes, il ne vous reste plus qu'à faire preuve de patience et à apprendre, pourquoi pas, en autodidacte. Sinon le plus simple et le plus rapide est de vous tourner vers l'installation de plusieurs instances de WordPress.

Si vraiment vous tenez à utiliser WordPress en mode multisite, faites appel à un professionnel ; il se chargera de la configuration de votre serveur et de l'installation de WordPress.

Les formats d'adresse

Le mode multisite permet de créer des sites selon deux formats d'adresse : les sous-domaines et les sous-répertoires. Le choix du format est à faire lors de la mise en place du mode multisite ; il sera très compliqué de le modifier en cours de route, d'où l'intérêt de bien y réfléchir…

Par exemple, si nous créons et nommons un site `site1` depuis WordPress, nous obtiendrons :

- soit *http://site1.monsite.fr/* pour les sous-domaines ;
- soit *http://monsite.fr/site1/* pour les sous-répertoires.

Le type sous-domaine est le plus utilisé, pour une raison principalement : le référencement. Un sous-domaine est considéré par les moteurs de recherche comme un site différent du domaine principal. On retrouve ce format sur les plates-formes publiques de blogs (type WordPress.com).

Dans le cas où vous utilisez WordPress pour un réseau de sites, il est préférable d'opter pour le type sous-répertoire pour faire profiter la globalité du domaine du référencement des sites.

Quand utiliser le mode multisite ?

Pour répondre à cette question, nous aurions pu créer un arbre décisionnel. Mais nous pouvons également détailler notre réponse par le biais d'une série de questions plus simples que voici.

Un ou plusieurs blogs ?

Il n'est pas interdit d'utiliser le mode multisite pour un seul et unique blog mais, honnêtement, quels avantages en tirerez-vous ? Ce mode est un peu plus gourmand en requêtes SQL et en ressources mémoires que le mode monosite. Pas de doute possible, si vous n'avez qu'un blog, utilisez de préférence WordPress tel quel.

Petite ou grande fréquentation ?

Vous possédez plusieurs blogs ? Des blogs connus ? Des blogs médiatiques ? Des blogs générant beaucoup de trafic ? Regrouper vos blogs sur une seule installation de WordPress n'est peut-être pas une bonne solution. Si vous avez par exemple trois blogs qui sont hébergés chez trois hébergeurs distincts, ou tout au moins trois comptes différents, la charge est

répartie directement chez vos hébergeurs sans que vous ayez à vous en soucier. Migrer vos trois blogs sur une seule installation de WordPress signifie que toute la charge sera centralisée sur un seul compte, une seule base de données… Autrement dit, votre hébergeur peut ne pas apprécier et vous demander par la même occasion d'aller voir ailleurs. Or un hébergement haut de gamme juste pour centraliser vos installations peut vous coûter bien plus cher que trois comptes distincts…

Quel hébergeur ?

Tous les hébergeurs ne sont pas compatibles avec le mode multisite. Un hébergeur comme Infomaniak, bien que réputé, ne gère qu'un seul domaine par compte. Ce mode n'est donc pas compatible avec ce type d'hébergeur. Généralement, le plus simple pour travailler est de prendre un hébergement virtualisé, voire un serveur dédié pour des gros besoins, ce qui implique des connaissances en administration serveur Linux et des frais supplémentaires...

Quel niveau technique ?

L'installation et la configuration de WordPress sont très accessibles. L'ajout du mapping de domaines sur les blogs d'une installation WordPress demande déjà un peu plus de compétences avec l'utilisation de sunrise, les modifications DNS, les vhost du serveur HTTP.

Enfin, la principale difficulté que l'on rencontre avec le mode multisite de WordPress vient des extensions non compatibles à 100 %. Par exemple, l'extension de génération de sitemap pour les moteurs de recherche, Google Sitemap Generator, n'est pas encore compatible avec ce mode à l'heure où nous écrivons ces lignes… L'intégration de ce mode dans WordPress est encore récente, et il faudra un temps d'adaptation pour que toutes les extensions prennent totalement en compte l'ensemble des fonctionnalités de ce nouveau mode. De fait, si vous avez des besoins par rapport à une extension en particulier, il vous faudra réaliser quelques modifications dans le code PHP… ou, pire, utiliser une extension alternative – si elle existe !

Conclusion

Utiliser le mode multisite de WordPress peut être pertinent lorsqu'on possède plusieurs blogs mais, avant de tout passer dans ce mode, il est indispensable de bien peser le pour et le contre et surtout d'être prêt à consacrer un peu plus de temps à la mise en place de la plate-forme.

Car, une fois installé et configuré, ce mode simplifie grandement la vie… Une seule plate-forme à mettre à jour, une connexion pour administrer ses différents blogs, une base de données, un compte hébergeur, tout cela fait gagner du temps…

WordPress en mode multisite est donc un bon choix… si vous avez les compétences techniques pour administrer la plate-forme !

14 Créer un réseau de sites avec WordPress

Configuration logicielle nécessaire

Serveur mutualisé ou serveur dédié ?

En fait, la question ne se pose pas vraiment. L'idéal pour le mode multisite est de disposer d'un serveur dédié, ou tout du moins virtualisé (VPS). Il y a en effet des prérequis assez contraignants à mettre en place pour ce mode ; de ce fait, l'idéal est d'avoir un accès complet sur le serveur.

Néanmoins, il est possible de faire fonctionner WordPress en mode multisite sur un hébergement dédié, si ce dernier répond aux prérequis détaillés dans la prochaine section. Avec les hébergeurs mutualisés les plus connus – OVH, 1&1, Amen –, il sera possible d'utiliser ce mode sans restriction, mais avec le format d'adresse sous-répertoire.

Pour le format sous-domaine, vous serez obligé de créer manuellement chaque sous-domaine dans l'interface d'administration de votre hébergement.

Prérequis

En mode normal comme en mode multisite, WordPress a besoin d'au moins PHP 4.3 et MySQL 4.1.2.

Cependant, de nombreuses extensions pour le mode multisite nécessitent les versions PHP 5 et MySQL 5. Par conséquent, essayez d'installer les versions les plus récentes de ces logiciels.

Pour la plupart des prérequis, les manipulations sont à effectuer depuis une console SSH ; cela nécessite une connaissance assez poussée des systèmes d'exploitation Linux.

Il est également possible de faire ces modifications depuis les interfaces d'administration de serveur telles que Webmin. Les résultats sont alors plus aléatoires.

Il n'y aura aucune commande à exécuter dans ce chapitre, pour la simple et bonne raison que les commandes diffèrent selon la famille Linux (Debian, Red Hat, Gentoo, etc.) de votre serveur web.

Si vous avez un responsable serveur, le plus simple sera encore de lui demander de rendre le serveur web conforme avec les prérequis suivants.

Réécriture des URL WordPress (serveur HTTP)

Si vous utilisez Apache, il est impératif d'activer le module rewrite ; le mode multisite ne fonctionnera pas sans.

Si vous utilisez des serveurs HTTP nouvelle génération tels que Lighttpd ou Nginx, il y a des règles pour WordPress MU à placer directement dans les fichiers de configuration. Vous pouvez les trouver sur le Codex : **http://codex.wordpress.org/Installing_WPMU#Rewriting_Rules_for_others_HTTP_Server**.

Gestion dynamique des sous-domaines (serveur HTTP, serveur DNS)

Cette modification est à effectuer uniquement pour le format d'adresse sous-domaine.

Pour permettre la création des sous-domaines sans avoir besoin de modifier la configuration serveur, il est nécessaire d'ajouter un wilcard sur le domaine de WordPress. Un wildcard correspond en français à la carte joker. Ce dernier va permettre à tous les sous-domaines d'être reconnus.

Dans notre exemple, le domaine de WordPress est *http://monsite.fr* et l'adresse IP de notre serveur dédié est 66.77.88.99.

Nous allons ajouter la ligne suivante dans la zone DNS de notre domaine. Généralement, les zones DNS sont contenues dans le dossier de configuration du serveur DNS Bind.

```
*.monsite.fr. IN A 66.77.88.99
```

Faites attention, le point à la fin de votre domaine est indispensable ! Si vous l'oubliez, votre configuration DNS sera corrompue.

Une fois la modification faite, mettez à jour le numéro de version de la zone DNS et relancez votre serveur DNS. Grâce à cette modification, tous les sous-domaines vont pointer sur l'adresse IP de serveur web.

N'oubliez pas non plus de modifier la configuration de serveur HTTP pour que ce dernier accepte tous les sous-domaines sur le vhost de WordPress.

Prenons le cas d'Apache. Généralement, le fichier httpd.conf contient les enregistrements suivants :

```
ServerName monsite.fr
ServerAlias www.monsite.fr
```

Nous allons modifier la directive ServerAlias et lui ajouter un wildcard. Le résultat attendu est :

```
ServerAlias www.monsite.fr *.monsite.fr
```

Grâce à ces modifications, tous les sous-domaines pointeront sur l'installation de WordPress.

Certains hébergeurs ont le wildard mis en place par défaut côté serveur, donc il ne vous reste qu'à modifier les DNS. Au contraire, certains hébergeurs (notamment les mutualisés) refusent ce type de modification...

Configuration et mise à jourInstallation de WordPress

L'installation de WordPress ne varie pas d'un iota : suivez les instructions données au Chapitre 2. C'est là l'un des grands apports de l'intégration de WPMU dans WordPress...

À noter qu'une fois les fichiers de WordPress en ligne vous pouvez soit installer le logiciel tel quel, l'utiliser en mode monosite pendant quelque temps, puis configurer le mode multisite ; soit modifier directement le fichier `wp-config.php` pour activer le mode multisite dès l'installation.

Une fois le mode multisite en place, il est fortement déconseillé de revenir en arrière...

Configuration

L'activation du mode multisite requiert quelques étapes, dont la première est on ne peut plus simple : elle ne requiert que l'ajout d'une ligne dans le fichier wp-config.php :

```
define('WP_ALLOW_MULTISITE', true);
```

Placez cette ligne à la fin du fichier (juste avant le commentaire vous déconseillant de modifier les lignes qui le suivent, voir Figure 14.01), puis rechargez WordPress (ou installez-le, selon où vous en êtes du processus), et voilà : vous êtes en mode multisite.

Les développeurs ont pris soin de rendre cette première étape simple afin de rester cohérents avec l'esprit de WordPress. Par chance, ils n'ont pas pris la décision de le rendre trop simple, comme l'aurait été la présence d'un bouton Activer le réseau de sites dans l'interface d'administration – le danger aurait été grand que les utilisateurs les moins aguerris cliquent dessus "pour voir" et se retrouvent à devoir gérer un réseau dont ils ne comprennent pas forcément les implications...

Figure 14.01

La ligne d'activation dans le fichier wp-config.php.

```
 * de thèmes se servent de WP_DEBUG dans leur environnement de
 * développement.
 */
define('WP_DEBUG', false);

define('WP_ALLOW_MULTISITE', true);

/* C'est tout, ne touchez pas à ce qui suit ! Bon blogging ! */

/** Chemin absolu vers le dossier de WordPress. */
if ( !defined('ABSPATH') )
    define('ABSPATH', dirname(__FILE__) . '/');
```

Attention, ce n'est pas terminé ! La constante que vous venez de placer dans `wp-config.php` ne fait qu'autoriser l'accès au mode multisite, il reste encore à le configurer ! Et cela se fait dans l'interface d'administration.

Quand la constante WP_ALLOW_MULTISITE est en place, rien ne change dans l'interface, à première vue. La seule différence, c'est l'apparition d'une page Réseau dans le menu Outils. Cette page, qui présente le titre "Créer un réseau de sites WordPress", prévient d'emblée : il vous faut d'abord désactiver toutes vos extensions avant de pouvoir mettre en place le réseau (voir Figure 14.02). Cette mesure préventive ne s'affiche que si vous avez effectivement des extensions activées...



Figure 14.02
Les extensions peuvent facilement perturber la mise en place du réseau.

Une fois les extensions désactivées, WordPress vous présente le formulaire de création du réseau (voir Figure 14.03). Celui-ci est très simple, et certains champs seront probablement déjà remplis en fonction de votre installation.

Le premier champ vous demande le format d'adresse des sites du réseau :

- **Sous-domaine.** Les adresses seront de la forme site1.exemple.fr, site2.exemple.fr, etc.
- **Sous-dossier.** Les adresses seront de la forme exemple.fr/site1, exemple.fr/site2, etc.

Si vous avez le choix, sélectionnez celui qui vous convient le mieux, mais restez conscient que vous ne pourrez pas revenir en arrière sans une réinstallation complète. Notez que chaque option implique des conséquences techniques : les sous-domaines requièrent un wildcard dans les champs DNS et la configuration Apache, tandis que les sous-dossiers nécessitent la présence de l'extension Apache mod_rewrite (déjà nécessaire pour avoir de beaux permaliens).

Viennent ensuite les détails du réseau :

- **Adresse du serveur.** En fait l'adresse web de votre réseau, tout simplement. Vous n'avez pas vraiment le choix, c'est celui où est installé WordPress.
- **Nom du réseau.** C'est le nom de baptême de votre réseau, qui sera présent sur toutes les correspondances aux membres du réseau. De fait, soyez concis.
- **Adresse de contact de l'administrateur.** Les courriers envoyés à l'équipe du réseau seront envoyés à cette adresse. N'indiquez donc pas forcément votre adresse personnelle...

Tous les champs sont remplis à votre convenance ? Validez le formulaire !

Si la configuration est incorrecte – message d'erreur ou page blanche –, le plus simple est de supprimer le fichier wp-config.php ainsi que les tables MySQL du site et de reprendre l'installation à zéro.

Si l'activation s'est bien déroulée, la page Réseau présente désormais une tout autre interface (voir Figure 14.04), présentant les dernières étapes de l'activation de votre réseau. Il vous faut les suivre scrupuleusement une à une – à commencer par une sauvegarde de vos fichiers wp-config.php et .htaccess (si ce dernier existe).

Figure 14.03

Le formulaire de création du réseau de sites WordPress.

Créer un réseau de sites WordPress

Aide

Bienvenue dans le processus d'installation du réseau !

Complétez le formulaire ci-dessous et vous serez prêt à créer un réseau de sites WordPress. Nous créerons les fichiers de configurations à l'étape suivante.

Note : Vérifiez bien que le module Apache `mod_rewrite` est installé, car il sera nécessaire à la fin de cette installation.

Si le module `mod_rewrite` est désactivé, contactez votre administrateur pour l'activer, ou lisez la documentation Apache officielle ou non, pour comprendre comment le mettre en place.

Adresse des sites dans votre réseau

Veuillez décider si vous voulez que les sites de votre installation WordPress utilisent des sous-domaines ou des sous-dossiers. **Ce réglage est définitif, vous ne pourrez pas revenir en arrière.**

Vous aurez besoin d'un enregistrement DNS générique (wildcard) pour permettre l'utilisation de la forme sous-domaine (hôte virtuel).

Sous-domaines — comme `site1.barbouillages.net` et `site2.barbouillages.net`

Sous-dossiers — comme `barbouillages.net/site1` et `barbouillages.net/site2`

Détails du réseau

Adresse du serveur — L'adresse internet de votre réseau sera `barbouillages.net`.

Nom du réseau — Réseau Barbouillages.net — Quel nom voulez-vous donner à votre réseau ?

Adresse de contact de l'administrateur — capuche@barbouillages.net — Votre adresse de contact.

Installer

Sans recopier le contenu de cette page, ces trois dernières actions sont les suivantes :

- Créer un dossier `/blogs.dir` dans le dossier `/wp-content` de votre installation de WordPress. C'est l'endroit où sont stockées les données des sites créés sur le réseau.
- Ajouter quelques lignes dans votre fichier wp-config.php. Les premières valident la mise en place de votre réseau, les secondes améliorent la sécurité de votre installation. Vous pouvez désormais effacer la ligne `define('WP_ALLOW_MULTISITE', true);`.
- Ajouter quelques lignes dans votre fichier .htaccess (qu'il faudra créer le cas échéant). Cela ajoute les règles de redirection nécessaires au bon fonctionnement des adresses web de votre réseau.

Ne modifiez pas ces chaînes, et n'oubliez pas une ligne, car chaque ligne compte. Si vous effacez par mégarde le contenu de `wp-config.php` ou de votre `.htaccess`, cette page réseau conservera toujours ces informations.

Les lignes ajoutées à `wp-config.php` pour améliorer la sécurité de votre réseau vous obligent à créer un nouveau cookie de connexion : reconnectez-vous à l'interface d'administration, avec les mêmes identifiant et mot de passe qu'avant.

En vous reconnectant, vous noterez un nouvel élément du menu de WordPress : "Super Admin". C'est d'ici que vous allez gérer votre réseau. Félicitations : le super admin, c'est vous !

Figure 14.04
Toutes les modifications nécessaires à la finalisation du réseau.



Quelques précisions importantes

Le mode multisite impose rapidement ses contraintes, qu'il faut respecter si l'on veut utiliser ce mode au maximum de ses possibilités.

Tout d'abord, vous devez placer WordPress à la racine du domaine. Il n'est pas recommandé de le mettre dans son propre sous-dossier, car cela interfère avec les blogs des membres.

Certains cas existent où il est tout simplement impossible de créer un réseau :

- Si la valeur du champ Adresse de WordPress est différente de celle du champ Adresse du site.
- Si le champ Adresse de WordPress utilise un port autre que 80 ou 443.

Par ailleurs, il existe trois cas où il n'est pas possible d'utiliser le format d'adresse en sous-domaines :

- WordPress est dans son propre dossier (pas à la racine du domaine).
- Le champ Adresse de WordPress contient "localhost".
- Le champ Adresse de WordPress contient "127.0.0.1".

De la même manière, il existe un cas où il est obligatoire d'utiliser le format sous-domaines : si l'installation date d'il y a plus d'un mois, l'horodatage du premier article en ligne faisant foi (ce afin de ne pas nuire à votre référencement).

Pas de www ?

Par défaut, WordPress supprime toujours la chaîne www depuis les adresses des sites utilisant votre site. Il sera toujours possible de visiter le site WordPress avec le préfixe www (par exemple *www.exemple.fr*), mais aucun lien contenu dans le site n'aura le préfixe www.

Ils seront tous sous la forme *http://exemple.fr*.

Ce choix s'explique tout simplement : en fait, les plates-formes de sites préfèrent ne pas utiliser le préfixe www pour disposer d'une adresse plus courte et plus simple.

Cela évite les adresses du type *www.site1.exemple.fr*.

Si vous voulez obtenir davantage d'informations sur les raisons pour lesquelles "www." n'est plus utile, vous pouvez consulter le site **no-www.org** (en anglais).

Le fichier de configuration

La constante `WPLANG` n'est pas très utile pour le mode multisite. En effet, la langue activée provient soit de la configuration de WordPress, soit de celle du blog.

Outre les constantes que l'outil de mise en place du réseau vous a demandé de placer dans ce fichier, deux constantes propres au mode multisite sont également disponibles en cas de besoin (voir Tableau 14.01).

Tableau 14.01 : Constantes de configuration spécifiques à WordPress multisite

Nom de la constante	Description
`SUNRISE`	Si cette constante est définie, WordPress tentera alors de charger le fichier wp-content/sunrise.php. L'utilité principale de cette constante est de remplacer la logique du fichier wp-settings.php par une version personnalisée permettant par exemple le mapping de domaine.
`NOBLOGREDIRECT`	Par défaut, lorsqu'on se trompe dans l'adresse d'un site de réseau WordPress, ce dernier ne redirige pas vers la page de création de blogs. Si on utilise WordPress multisite pour un réseau de blogs, ce n'est pas très pratique. Il suffit de définir une adresse dans cette constante pour changer la destination de la redirection.

Présentation de l'interface du Super Admin

Le mode multisite concentre ses pages d'administration dans le menu Super Admin (voir Figure 14.05). Le reste du menu fonctionne comme pour un blog WordPress normal et agit sur le site principal de la plate-forme.

Tableau de bord

Une fois que vous êtes connecté à WordPress en mode multisite, vous arrivez directement sur le tableau de bord du blog numéro 1.

Le site numéro 1, c'est le premier site créé par WordPress, c'est celui qui n'a pas de sous-domaine ni de sous-répertoire.

Ici, l'adresse du site numéro 1 est *http://exemple.fr/*.

Le menu Super Admin n'est accessible qu'aux administrateurs de la plate-forme WordPress. Un visiteur créant son propre site n'y a donc pas accès, bien qu'il soit administrateur de son propre site.

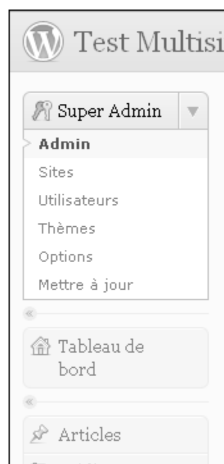


Figure 14.05
Le menu Super Admin, spécifique au mode multisite de WordPress.

Pour résumer, il y a deux types d'administrateurs :

- administrateur du site donnant les mêmes possibilités que le rôle WordPress ;
- administrateur du site permettant de paramétrer le mode multisite.

Ce menu contient six pages que nous allons détailler un peu plus loin dans ce chapitre.

Le dernier changement par rapport à WordPress est l'ajout d'une notion de quota pour chaque site. Ainsi dans le cadre Aujourd'hui, nous retrouvons l'espace disque disponible pour le site et l'espace disque utilisé ainsi que son pourcentage. Le mode multisite de WordPress permet en effet de gérer des quotas pour chaque site. Une valeur par défaut est définie pour la totalité de la plate-forme, mais il est possible de fixer cette valeur indépendamment pour chaque site.

Nous allons maintenant décortiquer le menu Super Admin.



Par défaut, WordPress vous indiquera en tête de page une notification : "Attention ! Le thème actuel accepte les images mises en avant (*via* miniatures). Vous devez activer l'envoi d'images depuis la page d'options pour que cela fonctionne." La page d'options étant bien longue, vous trouverez la case à cocher en question tout en bas, dans la section Réglages d'envoi des fichiers. Cochez la case Images et la notification disparaîtra.

Super Admin

La page d'accueil du Super Admin est minimaliste (voir Figure 14.06).

Elle affiche le nombre total de sites et d'utilisateurs présents sur votre installation, ainsi que deux formulaires pour chercher un utilisateur ou un site. Les deux liens sont des raccourcis vers d'autres écrans du mode multisite...



Figure 14.06
Page d'accueil du mode multisite.

Super Admin – Sites

La page Sites est déjà plus intéressante. Elle permet de lister les sites installés sur votre réseau WordPress (voir Figure 14.07). Vous avez la possibilité de filtrer le résultat avec une recherche sur le nom du site, l'adresse IP de création, ainsi que par l'ID du site.

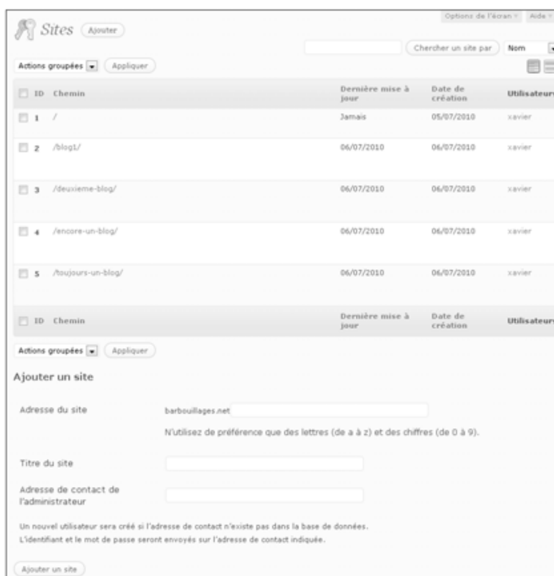


Figure 14.07
La page d'administration des sites du mode multisite de WordPress.

Les résultats peuvent être triés d'après différents critères ; pour cela il suffit de cliquer sur l'intitulé de la colonne du tableau de résultats. Pour inverser l'ordre de tri, il faut simplement cliquer une deuxième fois sur l'intitulé de la colonne.

Pour chaque site, vous avez à votre disposition des informations et différentes actions.

Vous retrouvez dans l'ordre :

- **Une case à cocher.** Elle permet d'appliquer une action sur plusieurs sites en même temps.
- **ID.** Correspond à l'identifiant du site. Ce dernier est un nombre et il est unique. Il s'incrémente automatiquement à chaque création de sites.
- **Chemin/Sous-domaine.** Le nom de la deuxième colonne dépend du format d'adresse choisi :
  - Si vous avez sélectionné sous-domaine, l'intitulé sera sous-domaine.
  - Sinon ce sera chemin pour le format sous-répertoire.

  Cette colonne correspond au nom du site choisi par l'utilisateur lors de la création.
- **Dernière mise à jour.** Correspond à la date de la dernière modification ou ajout d'un article, page ou lien sur le site.
- **Date de création.** Comme son nom l'indique.
- **Utilisateurs.** Cette colonne contient la liste des utilisateurs appartenant au site. La liste est tronquée à partir de dix utilisateurs au maximum.

Au survol de chaque ligne apparaît une liste d'actions. Les actions sont multiples. Elles permettent de modifier toutes les options du site depuis une unique page, d'accéder au tableau de bord de ce site, la désactivation et réactivation du site, mais également l'archivage du site, de déclarer un site comme étant du spam (site indésirable), de supprimer le site et enfin de l'afficher. Le site principal ne dispose que des liens Modifier, Administration et Afficher.

Si un site est désactivé, archivé ou classé comme indésirable, son contenu ne sera alors plus accessible au grand public, et son interface d'administration sera elle aussi désactivée. Une fois qu'un site est supprimé, il est impossible de le restaurer, son contenu est définitivement effacé. Faites donc bien attention au lien Supprimer et faites régulièrement des sauvegardes de votre installation.

Néanmoins, pour nuancer notre propos, précisons qu'une page de confirmation existe pour chacune des actions sur les sites.

La deuxième fonctionnalité de la page Sites permet la création d'un site depuis l'interface d'administration. Pour cela, cette page propose un formulaire à trois champs (Adresse du site, Titre du site, et Adresse de contact de l'administrateur).

Il faut savoir que si l'adresse de contact n'existe pas dans la base de données, un nouvel utilisateur est créé. L'identifiant et le mot de passe seront alors envoyés sur l'adresse de contact spécifiée.

Mais revenons sur le détail de l'action Modifier. Une fois sur la page d'édition du site, nous retrouvons cinq blocs d'options.

Dans la colonne de gauche, dans un premier bloc, nous avons le paramétrage du site présent dans la base de données. Majoritairement, ce sont les informations de la liste des sites vue précédemment.

Le deuxième bloc de la colonne de gauche affiche pour sa part la liste entière des options du site. Ce sont les options de la configuration du site WordPress, celles qu'on retrouve dans l'onglet Réglages. Elles comprennent généralement l'ensemble des paramétrages des extensions. Il est déconseillé d'y toucher sans savoir exactement ce que vous faites...

La colonne de droite propose trois blocs. Le premier permet de gérer individuellement les utilisateurs du site. Le deuxième permet de gérer l'ajout d'utilisateurs sur le site, ainsi que l'attribution des rôles WordPress (il faut que l'utilisateur existe déjà pour que cela fonctionne). Le troisième bloc concerne le quota du site ; il est ainsi possible de définir le quota pour un site en particulier.

Super Admin – Utilisateurs

Cette deuxième page est plus conventionnelle sur la forme que celles qu'on retrouve dans WordPress habituellement. Elle permet de lister tous les utilisateurs du réseau WordPress (voir Figure 14.08). Notez la présence d'une colonne Sites, qui permet de répertorier les sites auxquels l'utilisateur contribue.

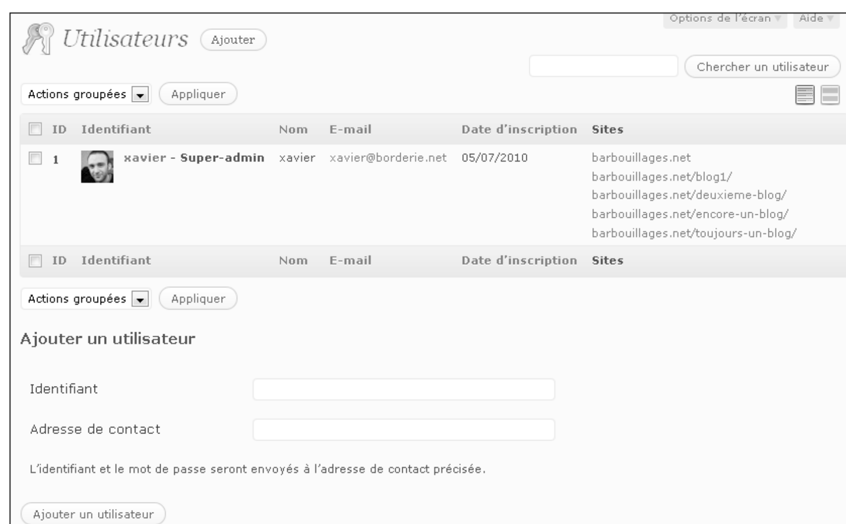


Figure 14.08
La page d'administration des utilisateurs du réseau.

Vous disposez ici des mêmes possibilités de tri des résultats, c'est le même mode de fonctionnement que la page Sites vue ci-dessus.

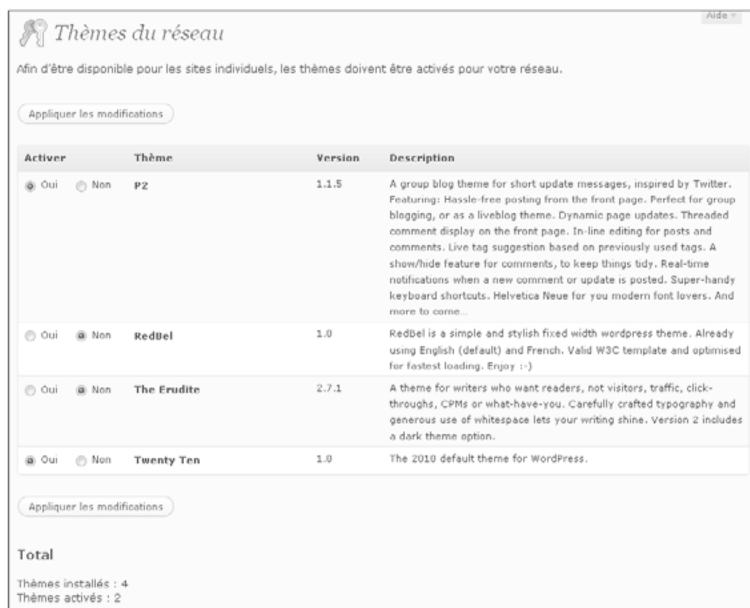
Vous avez ainsi la possibilité d'appliquer une action sur plusieurs utilisateurs ; vous pouvez alors supprimer par lots, marquer plusieurs utilisateurs comme indésirables ou légitimes, en un seul clic !

Enfin, un peu comme sur la page Sites, vous avez un formulaire d'ajout d'utilisateurs. Il suffit d'entrer l'identifiant et l'adresse e-mail, un mot de passe sera automatiquement envoyé à l'adresse de contact spécifiée.

Super Admin – Thèmes

Contrairement à WordPress en mode monosite, le mode multisite ne liste pas l'intégralité des thèmes dans le sélecteur de thèmes de chaque site. Il y a en effet une étape intermédiaire lors de l'installation d'un thème. Pour ajouter un thème dans le réseau de sites, il faut télécharger ses fichiers dans votre installation principale de WordPress (*via* Apparence > Thèmes > Installer des thèmes), puis le rendre actif grâce à cette page Thèmes de la partie Super Admin (voir Figure 14.09).

Figure 14.9
La page d'administration des thèmes du réseau WordPress.



Si vous souhaitez mettre à disposition un thème uniquement pour un site en particulier, il vous suffit de laisser le thème sur l'état non actif. Rectifiez ensuite les détails du site *via* l'action Modifier de la page Sites.

Vous avez ainsi la possibilité d'activer manuellement les thèmes inactifs.

Super Admin – Options

Cette page contient le paramétrage spécifique au mode multisite de WordPress (voir Figure 14.10). Ces options sont modifiables uniquement par les administrateurs du réseau.

Vous pouvez paramétrer de nombreux aspects du réseau, parmi lesquels :

- Le nom du site et l'adresse de contact utilisée lors de l'envoi de messages par le réseau.
- L'autorisation ou non de création de sites, d'utilisateurs ou bien des deux.
- Le message de bienvenue envoyé aux utilisateurs.
- Le premier article et le premier commentaire créés lors de l'ajout d'un site, avec leurs auteurs respectifs.

- Les restrictions, avec les noms de sites bannis, les domaines d'adresses e-mail autorisés et interdits.
- Les options liées au quota des sites, la quantité d'espace disque allouée par site, les types de fichiers acceptés, la taille maximale des fichiers envoyés.
- Le flux RSS qui s'affichera dans le tableau de bord de chaque site. C'est très pratique pour communiquer avec vos utilisateurs sur l'actualité du service par exemple.
- Les types de fichiers autorisés en envoi.

Figure 14.10
La page de configuration du mode multisite.

Vous pouvez également définir la liste des administrateurs du site et la langue par défaut des sites créés.

Enfin, vous avez la possibilité d'activer ou non la page d'extensions de WordPress dans l'interface d'administration des sites de WordPress.

Super Admin – Mettre à jour

Lorsque vous mettez à jour WordPress, il est parfois nécessaire de le faire aussi pour la base de données de chaque site. Vous avez deux moyens d'y arriver : la méthode manuelle, qui consiste à consulter l'interface d'administration et à procéder à la mise à jour, et la méthode automatique qui peut être lancée depuis la page Mettre à jour du menu Super Admin. Il est ainsi possible d'automatiser la mise à jour sur tous les sites en un clic !

Cette page nécessite l'activation de JavaScript pour automatiser le rechargement de la page. WordPress procède à la mise à jour par groupe de cinq sites, il relance ensuite la page et passe aux sites suivants, et ainsi de suite.

Il fonctionne ainsi simplement pour ne pas planter le processus PHP par une exécution trop gourmande et trop longue, dans le cadre d'une plate-forme regroupant beaucoup de sites.

Réglages – Général

Les deux champs Adresse du site et Adresse du WordPress disparaissent de la page ; ils ne sont plus utiles étant donné que l'adresse du site est définie à la création de celui-ci.

Un sélecteur permet de choisir parmi les langues disponibles. Sur la version française, vous retrouvez donc deux langues : anglais et français.

Outils – Supprimer le site

Dans une orientation plate-forme publique de site avec WordPress en mode multisite, étant donné que les utilisateurs ont la possibilité de créer leur propre site, il est obligatoire de leur donner aussi la possibilité de fermer leur site (voir Figure 14.11). Cette page n'est pas disponible pour le site principal, qui par définition ne peut pas être effacé.

Figure 14.11
Supprimer son site depuis WordPress.

Tout cela est possible depuis cette page. Le processus d'effacement se déroule en deux temps : dans un premier temps, vous devez cocher la case assurant que vous êtes conscient qu'il sera impossible de restaurer le site une fois qu'il sera supprimé, puis dans un deuxième temps vous devez valider le formulaire.

WordPress envoie alors un e-mail de confirmation, qui contient le lien validant la suppression. Une fois le site supprimé, même l'administrateur n'est plus en mesure de le restaurer : les données de la base sont effacées ainsi que les fichiers du site.

15 Spécificités du développement en mode multisite

Dans ce chapitre, nous allons nous intéresser aux caractéristiques du développement avec un réseau WordPress.

Le réseau et les sites sous WordPress en mode multisite

Depuis WordPress 3.0, la notion de blog disparaît des textes du logiciel : désormais, on crée un site avec WordPress. Que celui-ci soit un blog ou non, là n'est pas l'important : cela reste un site.

De fait, cela va à l'encontre de l'acception précédente : avec WordPress MU, il y avait le site principal, qui contenait des blogs. Maintenant que les blogs sont considérés comme des sites à part entière, le site principal dispose d'une nouvelle appellation : le réseau. C'est une différenciation primordiale lors du développement d'extensions ou de thèmes consacrés au mode multisite.

Un réseau est un domaine, une plate-forme. Les sites, eux, appartiennent obligatoirement à un réseau.

Par exemple :

- Le réseau 1 a pour thème les voitures, le domaine utilisé est *blogs-voitures.fr*.
- Le réseau 2 a pour thème les motos, le domaine utilisé est *blogs-motos.fr*.

Nous souhaitons proposer des sitess avec soit le domaine *blogs-voitures.fr*, soit le domaine *blogs-motos.fr*. Dans WordPress, ces deux domaines correspondent à deux réseaux différents. Chacun dispose de sa propre configuration. Par défaut, l'onglet Super Admin contient les sites, les utilisateurs, le paramétrage du seul réseau existant sur votre installation.

WordPress ne propose pas directement une interface permettant de lister et de gérer plusieurs sites. Vous pouvez cependant trouver une extension nommée WP Multi Network, qui ajoute cette fonctionnalité.

Attention, si vous désirez fixer un domaine précis sur un site, créer un réseau est une mauvaise solution. C'est un usage courant car il est assez simple, mais cela peut vite se révéler problématique car chaque réseau dispose de sa propre configuration. Concrètement, cela alourdit grandement l'administration de la plate-forme. Idéalement, vous devez utiliser une extension permettant le mapping d'un domaine sur un site.

Ce qui signifie que vous devez définir un domaine pour un site existant. En pratique, cela permet à l'utilisateur final de disposer de sa propre adresse. Par exemple, au lieu d'avoir *http://exemple.fr/jules/*, l'utilisateur pourra acheter le domaine *http://lesitedejules.fr/* et le fixer sur son site.

À l'heure actuelle, il y a un mu-plugin qui prend tout cela en charge. Il est proposé par Donncha O'Caoimh, qui était le mainteneur officiel de WordPress MU (**http://wordpress.org/extend/plugins/wordpress-mu-domain-mapping/**).

La base de données de WordPress en mode multisite

Depuis que WordPress est passé en mode multisite, la base de données se présente différemment : il y a en effet huit tables par site, formatées de la façon suivante : `wp_<n>_nomdelatable` (`n` étant l'ID du blog).

En plus des tables de contenu de WordPress, on retrouve neuf tables communes à tous les blogs.

Six tables sont créées lors de l'installation du mode multisite :

- `wp_blogs` ;
- `wp_blog_versions` ;
- `wp_registration_log` ;
- `wp_signups` ;
- `wp_site` ;
- `wp_sitemeta`.

Tandis que les deux tables liées à la gestion utilisateur reprennent la structure du mode monosite, mais ne sont pas communes à tous les sites :

- `wp_usermeta` ;
- `wp_users`.

Ces tables permettent de savoir quel utilisateur fait partie de quels sites, et quels sont ses droits sur le site en question. Les six tables spécifiques au mode multisite sont détaillées plus loin dans ce chapitre. Avec la fusion de WPMU dans WordPress, certains fichiers de l'installation de WP ne sont utiles qu'au mode multisite.

- `wp-activate.php` ;
- `wp-signup.php` ;
- `wp-admin/ms-admin.php` ;
- `wp-admin/ms-delete-site.php` ;
- `wp-admin/ms-edit.php` ;
- `wp-admin/ms-options.php` ;
- `wp-admin/ms-sites.php` ;
- `wp-admin/ms-themes.php` ;
- `wp-admin/ms-upgrade-network.php` ;
- `wp-admin/ms-users.php` ;
- `wp-admin/my-sites.php` ;

- `wp-admin/network.php` ;
- `wp-admin/includes/ms.php` ;
- `wp-admin/includes/ms-deprecated.php` ;
- `wp-includes/ms-blogs.php` ;
- `wp-includes/ms-default-constants.php` ;
- `wp-includes/ms-default-filters.php` ;
- `wp-includes/ms-deprecated.php` ;
- `wp-includes/ms-files.php` ;
- `wp-includes/ms-functions.php` ;
- `wp-includes/ms-load.php` ;
- `wp-includes/ms-settings.php`.

Ces fichiers seront également examinés en détail dans ce chapitre.

Enfin, WordPress MU contient un dossier mu-plugins en plus du classique plugin. Les différences entre les deux seront abordées ultérieurement dans ce chapitre.

Architecture

La question de l'architecture de la base de données a souvent donné lieu à des débats enflammés dans la blogosphère à l'époque de WordPress MU.

En effet, le mode multisite a la particularité de créer pour chaque site neuf tables dans la base de données MySQL. Faites le calcul : vous obtenez 9 000 tables pour un réseau de 1 000 sites.

Pourquoi alors ne pas avoir attribué l'ID du blog pour chaque type de contenu ? Nous nous serions retrouvés avec neuf tables pour 1 000 sites.

Continuons dans cette hypothèse et disons que chaque blog possède en moyenne 100 articles. La table Articles compterait un total de 100 000 articles.

Vertigineux non ?

Le cas Lyceum

Ce choix monolithique de tables, c'est le chemin qu'a privilégié le créateur du projet Lyceum. Un fork basé sur WordPress MU de l'époque, géré par Fred Stutzman et John Joseph Bachir, privilégiant le nombre de tables au nombre d'entrées par table.

Quoi qu'on en dise, ce choix d'architecture est mauvais, cela pour plusieurs raisons.

- La sécurité et l'intégrité des données posent problème. Si la table MySQL est corrompue, toutes les données le sont aussi.
- La compatibilité des extensions WordPress est cassée. En modifiant le schéma SQL des tables de WordPress, Lyceum n'est pas compatible avec les extensions de WordPress.
- La montée en charge et les performances sont problématiques. WordPress utilise la base de données MySQL. Bien qu'elle soit énormément employée dans le monde Open Source, MySQL est très loin de valoir les performances des bases de données comme Oracle, pour ne citer qu'elle.

Comment répartir la charge si l'intégralité des données est sur huit tables ? C'est une question sans réponse…

Lyceum était un projet voué à l'échec, et son principal développeur a d'ailleurs jeté l'éponge en mars 2009. Avec l'arrivée de WordPress 3.0 et l'excellente qualité des outils système distribués de gestion de code, comme Git ou Mercurial, John Joseph Bachir espère pouvoir relancer le projet un jour ou l'autre si d'autres le rejoignent dans l'aventure, mais gageons qu'il y a peu d'espoir que cela se fasse...

Le choix du mode multisite

Revenons au mode multisite de WordPress, et à ses neuf tables par site. Quel est l'intérêt de conserver autant de tables ?

- La compatibilité des extensions WordPress est préservée en mode multisite. Cela n'est vrai que si l'extension est développée correctement – ce qui est le cas de la majorité des extensions populaires.
- L'intégrité des données est simple à gérer. Si un site plante pour une raison X ou Y, ce sont ses propres données qui sont perdues, pas celles du site voisin… Il suffit ensuite de restaurer les données depuis une sauvegarde et le problème est réglé.
- La montée en charge est possible par différents moyens. Sans entrer dans les détails, sachez qu'on peut répartir les tables des sites sur différentes grappes de serveurs MySQL.

Il existe à ce sujet une classe de connexion appelée `HyperBD`, conçue par Automattic et permettant de gérer la répartition, la réplication des données, ainsi que le "failover".

Cette classe de connexion ne sera pas abordée dans ce livre. Vous pouvez trouver une documentation succincte et en anglais sur le Codex de WordPress : **http://codex.wordpress.org/HyperDB**.

Les tables du réseau WordPress

Le Tableau 15.01 fait un descriptif des tables ajoutées par WordPress en mode multisite aux tables principales.

Tableau 15.01 : Les tables WordPress en mode multisite

Nom des tables	Description
`wp_blogs`	Contient les informations principales de chaque site : l'ID, l'adresse du site, la date de création et de dernière mise à jour et le statut (public, spam, archive).
`wp_blog_versions`	Table dépréciée.
`wp_registration_log`	Contient l'historique des inscriptions sur le réseau, aussi bien les sites que les utilisateurs. Elle conserve les données à vie, même après l'effacement du blog.
`wp_signups`	Table de données temporaires qui stocke les inscriptions de sites non validés. L'inscription publique se passe en deux temps avec une validation.
`wp_site`	Contient la liste de site renfermant des informations : ID, domaine et chemin (path).
`wp_sitecategories`	Cette table porte mal son nom depuis WordPress 2.3 ; elle comprend l'intégralité des termes utilisés dans les sites. Cette table est mise à disposition des extensions ; elle ne propose aucune administration par défaut.
`wp_sitemeta`	Contient le paramétrage du réseau, ce qu'on retrouve dans les options du menu Super Admin.
`wp_users`	Cette table existe par défaut, mais le mode multisite ajoute deux champs pour chaque utilisateur.

Les fichiers supplémentaires

Les fichiers supplémentaires dédiés au mode multisite sont détaillés au Tableau 15.02.

Tableau 15.02 : Les fichiers de WordPress en mode multisite

Fichier	Description
`wp-activate.php`	C'est la page d'activation des comptes utilisateur du réseau. Lorsqu'un utilisateur crée son site ou son identifiant, il reçoit un lien de confirmation par e-mail ; c'est sur cette page qu'a lieu le processus d'activation.
`wp-signup.php`	Page de création de sites ou d'utilisateurs.
`wp-includes/ms-settings.php`	Véritable cœur du mode multisite, ce fichier décortique l'adresse de votre navigateur pour déterminer le site et le site sur lesquels vous naviguez. Il charge ainsi la bonne configuration de la base de données du site que vous visitez.
`wp-admin/ms-admin.php`	Interface d'administration. Tableau de bord du mode multisite.
`wp-admin/ms-edit.php`	C'est ici qu'ont lieu toutes les actions des pages du mode multisite. C'est une page à décortiquer dans le cadre d'un développement des extensions interagissant avec l'administration du réseau, pour bien comprendre leur fonctionnement.
`wp-admin/ms-options.php`	Interface d'administration. Options du réseau.
`wp-admin/ms-sites.php`	Interface d'administration. Liste des sites.
`wp-admin/ms-themes.php`	Interface d'administration. Liste des thèmes actifs ou non.
`wp-admin/ms-upgrade-network.php`	Interface d'administration. Mise à jour de tous les sites.

Fichier	Description
`wp-admin/ms-users.php`	Interface d'administration. Liste des utilisateurs.
`wp-admin/my-sites.php`	Interface d'administration. Liste des sites de l'utilisateur courant.
`wp-admin/includes/ms.php`	Ce fichier contient les fonctions PHP relatives au mode multisite. Elles ne sont utilisées et disponibles que depuis l'interface d'administration.
`wp-admin/includes/ms-deprecated.php`	Ce fichier sert de cimetière aux fonctions obsolètes de WordPress MU, en cas de besoin. Elles seront à terme définitivement enlevées.
`wp-includes/ms-blogs.php`	Contient les fonctions permettant au site d'interagir avec ses tables et ses données.
`wp-includes/ms-default-constants.php`	Définit les constantes et variables globales du mode multisite. Elles peuvent être remplacées par les valeurs définies dans wp-config.php.
`wp-includes/ms-default-filters.php`	Définit la plupart des filtres et crochets relatifs au mode multisite.
`wp-includes/ms-deprecated.php`	Un autre cimetière à fonctions obsolètes...
`wp-includes/ms-files.php`	Autorise les fichiers envoyés depuis le gestionnaire de médias de WordPress. Prenons l'exemple de l'envoi d'une image et son insertion dans un article. Contrairement à WordPress, qui va mettre l'adresse définitive de l'image (*http://monsite.fr/wp-content /uploads/monimage.jpg*), WordPress MU va construire une adresse intermédiaire ne permettant pas de connaître l'arborescence de l'image. Cela donnera une adresse du type *http://wp-mu.fr/files /monimage.jpg*. Cette adresse est rewritée directement par le fichier .htaccess de WordPress. De ce fait, on ne connaît pas l'emplacement exact de l'image, ce qui est pratique lorsque le stockage des fichiers n'est pas fait sur le serveur web.
`wp-includes/ms-functions.php`	Ce fichier contient la plupart des fonctions dédiées au mode multisite. Ce fichier est lancé durant l'initialisation et les fonctions sont disponibles aussi bien côté utilisateur que côté administration.
`wp-includes/ms-load.php`	Les fonctions nécessaires au lancement du mode multisite.
`wp-includes/ms-settings.php`	Définit certaines variables générales.

Un effort de documentation a été fait sur le Codex (**http://codex.wordpress.org/WPMU_Functions**), en ce qui concerne le fichier ms-functions.php (auparavant wpmu-functions.php), pour détailler l'usage et l'utilité de chaque fonction (uniquement celles qui sont spécifiques à WordPress MU). La mise à jour pour la version 3.0 se fait attendre...

Sunrise, quèsaco ?

Nous avons vu précédemment que nous pouvions définir une constante `SUNRISE` dans le fichier de configuration. Cette constante, si elle est définie, permet de charger le fichier wp-content/sunrise.php. Ce fichier est lancé juste avant le fichier ms-settings.php, ce qui permet de remplacer la logique par défaut du mode multisite par la vôtre.

Par exemple, il est possible de charger un fichier sunrise.php permettant le mapping de domaine. Vous pouvez également utiliser sunrise.php dans le cadre d'une répartition des blogs sur différents serveurs MySQL, etc.

Les mu-plugins et les extensions

WordPress peut utiliser deux dossiers pour les extensions : plugins et mu-plugins. Ce second dossier était à l'origine spécifique à WordPress MU, mais son utilité est toujours réelle maintenant que la fusion est faite, il a donc été conservé et son "mu" ne signifie plus "multi-user" mais "must-use" (utilisation obligatoire). Notez qu'il n'existe pas par défaut, vous devez le créer à la main...

Les mu-plugins sont lancés avant les extensions classiques de WordPress dans l'initialisation du mode multisite.

Le dossier plugins fonctionne normalement : le logiciel parcourt les fichiers et les dossiers de premier niveau et propose les extensions depuis la page Extensions de l'interface d'administration.

Le dossier mu-plugins fonctionne différemment. Une fois en mode multisite, WordPress lit uniquement les fichiers PHP contenus dans le dossier, il ne parcourt pas les sous-dossiers. Il est souvent nécessaire de créer un fichier PHP qui va faire appel à l'extension pour conserver les extensions dans leur dossier original.

Par exemple, pour inclure Simple Tags dans les mu-plugins, nous plaçons le dossier simple-tags dans mu-plugins et nous créons un fichier PHP directement dans le dossier mu-plugins avec le nom simple-tags-mu.php.

Ce fichier contiendra le code suivant :

```
<?php
@include( dirname(__FILE__) .  '/simple-tags/simple-tags.php') ;
?>
```

En règle générale, il convient de placer les extensions indispensables à la plate-forme dans le dossier mu-plugins, alors que les extensions optionnelles seront stockées dans le dossier plugins pour permettre à l'utilisateur de lancer ou non les plugins dont il a besoin.

Précautions à prendre lors du développement d'extensions

En toute logique, une extension bien développée pour WordPress sera compatible avec le mode multisite. Mais ce n'est malheureusement pas le cas de toutes les extensions. Nous allons voir les deux erreurs les plus fréquentes qui cassent cette compatibilité.

La variable *$table_prefix*

La première erreur consiste à utiliser la variable `global $table_prefix`. Cette dernière est dépréciée depuis la version 2.1 de WordPress, mais cela n'empêche pas certains auteurs d'extensions de l'employer, ce qui est déconseillé.

Il faut plutôt utiliser la variable `global $wpdb`, qui est l'objet de la classe de connexion à la base de données. Il est préférable d'employer un maximum les données de cette dernière, car vous avez la certitude que les données y sont correctes.

Donc vous devez remplacer `global $table_prefix` par `global $wpdb`, et vous obtenez la valeur du préfixe de table comme ceci : `$wpdb->table_prefix`.

Les chemins de fichiers en dur

Le second problème concerne l'utilisation de chemin en dur dans l'extension. Il n'est pas rare de trouver les fonctions `include()` et/ou `require()` qui font appel à des fichiers avec des chemins à moitié écrits en dur.

Par exemple, lorsqu'une extension souhaite charger le code d'un widget, l'appel suivant est souvent fait :

```
Include ( ABSPATH . 'wp-content/plugins/mon-plugin/widget.php' );
```

Il est préférable de remplacer cet appel par :

```
include ( dirname(__FILE__) . '/widget.php' ) ;
```

La première ligne ne fonctionnera pas dans les mu-plugins, alors que la deuxième ligne fonctionnera à coup sûr, quel que soit le dossier du plugin.

Enfin, le choix des plugins pour un réseau WordPress fait entrer en jeu plus de critères qu'un simple site WordPress. L'aspect performances est beaucoup plus important dans le mode multisite, pour permettre une montée en charge. Ce n'est cependant pas le seul critère ; il faut savoir que toutes les extensions WordPress monosite ne sont pas forcément intéressantes pour un réseau WordPress.

En fait, tout dépend de l'usage du réseau. Dans le cadre d'un réseau de sites personnels, il n'y a pas de grandes différences avec une utilisation normale. Dans le cadre d'une plate-forme grand public, les critères de facilité d'utilisation, d'internationalisation de l'extension et de réponses à un besoin de la communauté entrent en jeu.

Spécificité pour les thèmes multisite

Du côté des thèmes conçus pour un réseau WordPress, il n'y a strictement aucune différence avec les thèmes classiques. Ils font appel tous les deux aux mêmes fonctions. De ce fait, un thème WordPress sera compatible avec un réseau WordPress.

Cependant, faites attention aux thèmes de type magazine, car ces derniers sont généralement très mal développés.

Le plus souvent, il y a des ID de catégories définis directement dans le code du thème, ce qui est incompatible avec une utilisation du mode multisite. Sans oublier que ces thèmes utilisent généralement les champs personnalisés de WordPress pour gérer les images, fonction qui est beaucoup plus compliquée à exploiter en mode multisite (pas d'accès FTP par exemple).

Pour les fonctionnalités propres au mode multisite, comme les derniers articles de la plateforme, les derniers sites, ou le lien de création de sites, il y avait un thème Home livré avec WordPress MU. Il contenait les fonctions à utiliser et était un très bon exemple pour intégrer des fonctionnalités MU dans un thème.

Malheureusement, point d'équivalent dans Twenty Ten, le nouveau thème par défaut de WordPress. Il vous faudra donc mettre la main à la pâte et réaliser cette page vous-même. Voici un exemple que nous vous proposons, inspiré de ce thème Home si utile (voir Figure 15.01).

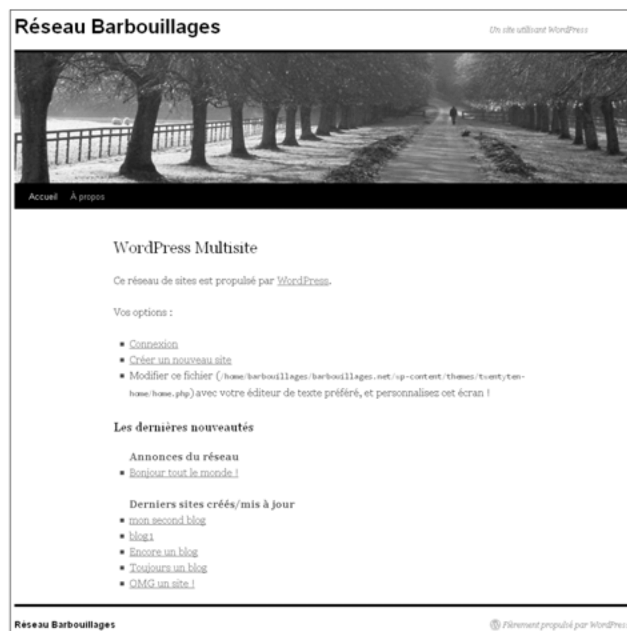


Figure 15.01
Une page par défaut pour votre réseau.

Vous avez certainement lu la partie de ce livre dédiée à la création de thème et connaissez donc la hiérarchie des modèles de WordPress. Si vous ne l'avez pas lue, faites-le impérativement avant de modifier un thème sans vraiment savoir ce que vous faites...

Le fichier index.php de Twenty Ten est donc celui qui affiche les derniers articles du site principal. Nous voulons afficher une page d'accueil, donc, pour supplanter index.php, nous allons créer un fichier home.php.

Mais nous voulons également que notre fichier ne disparaisse pas à la prochaine mise à jour, ce qui sera le cas si nous le mettons directement dans le dossier de Twenty Ten. La solution consiste à créer un thème enfant basé sur ce thème.

Nous allons donc ici créer le thème "Twenty Ten with Home", qui ne fera qu'ajouter la page home.php aux fichiers existants de Twenty Ten. Comme vous le savez sûrement, nous devons alors créer un nouveau dossier pour héberger nos fichiers : ce sera `/wp-content/themes/twentyten-home`.

Ensuite, il nous faut au minimum un fichier style.css référençant le dossier du thème parent, avec le mot-clef Template. Ce fichier ne fera qu'importer la feuille de style originale de Twenty Ten :

```
/*
Theme Name: Twenty Ten with Home
Description: This TwentyTen child-theme adds a home.php page to the theme,
nothing more, nothing less.
Author: Xavier Borderie
Version: 1.0
Template: twentyten
*/
@import url("../twentyten/style.css");
```

Enfin, créons le seul fichier qui différencie ce thème enfant de son original : home.php.

Voici le code complet du fichier :

```
<?php
get_header(); ?>

<div class="one-column">
  <div id="content">

  <h2>WordPress Multisite</h2>
  <p>Ce réseau de sites est propulsé par <a href="http://wordpress.
org/">WordPress</a>.</p>
  <p>Vos options :
    <ul>
      <?php wp_register(); ?>
      <li> <?php wp_loginout(); ?></li>
      <li> <a href="wp-signup.php">Créer un nouveau site</a></li>
      <li> Modifier ce fichier (<code><?php echo __FILE__; ?></code>) avec votre
éditeur de texte préféré, et personnalisez cet écran !</li>
    </ul>
  </p>
  <h3>Les dernières nouveautés</h3>
    <ul>
      <strong>Annonces du réseau</strong>

<?php
query_posts( 'showposts=7' );
if ( have_posts() ) : ?><?php while ( have_posts() ) : the_post(); ?>
      <li>
        <a href="<?php the_permalink(); ?>" rel="bookmark" title="Lien
permanent vers <?php the_title(); ?>"><?php the_title();?> </a>
      </li>
<?php endwhile; ?><?php endif; ?>
    </ul>
<?php
$blogs = get_last_updated();
if ( is_array( $blogs ) ) {
  ?>
    <ul>
      <strong>Derniers sites créés/mis à jour</strong>
  <?php foreach ( $blogs as $details ) { ?>
      <li>
```

```
        <a href="http://<?php echo $details[ 'domain' ] . $details[ 'path' ]
?>"><?php echo get_blog_option( $details[ 'blog_id' ], 'blogname' ); ?></a>
      </li><?php
  }
  ?>
    </ul>
  <?php
}
?>
  </div>
</div>
```

Une fois les fichiers en place dans le dossier /twentyten-home, activez ce thème pour l'ensemble du réseau dans la page Super Admin > Thèmes. Puis activez-le pour le site principal *via* la page Apparence > Thèmes. Rechargez l'accueil de votre réseau, et vous voilà avec une première page d'accueil. À vous de la modifier selon vos besoins.

Partie V

LE CAMPUS

Les autres projets de la WordPress Foundation

16 BuddyPress – la face sociale de WordPress

Présentation

BuddyPress est un projet ayant pour vocation de transformer une installation basique de WordPress en quelque chose qui ressemble à un outil communautaire (voir Figure 16.01), voire un outil de réseau social.

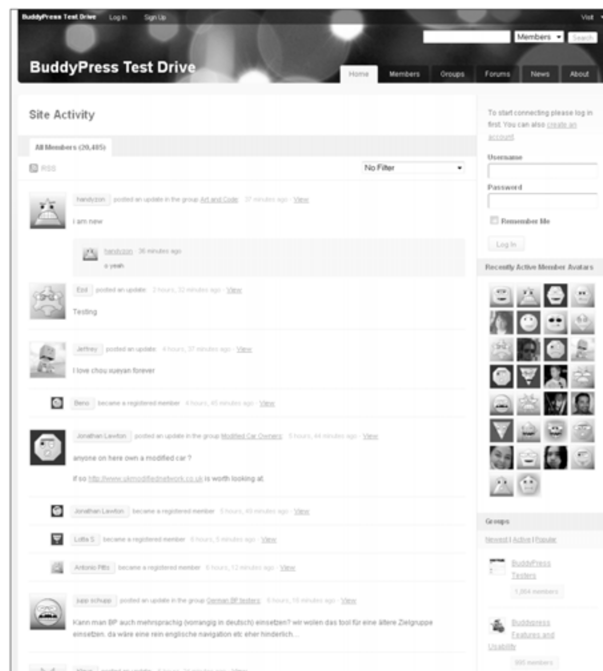


Figure 16.01
Le thème par défaut de BuddyPress.

Pour cela, BuddyPress ajoute des fonctionnalités propres aux réseaux sociaux, telles qu'un profil utilisateur complet, une messagerie et une gestion de groupes, tout en conservant les fonctionnalités propres à WordPress, à commencer par la création de blogs.

BuddyPress fait passer les utilisateurs au premier plan au détriment des blogs, ces derniers devenant une fonctionnalité parmi les autres. Beaucoup de gens considèrent BuddyPress comme l'avenir de WordPress.com et dans une moindre mesure de la version multisite de WordPress !

Historique

Le développeur de BuddyPress s'appelle Andy Peatling. Il a développé en 2007 une première version de BuddyPress pour l'organisation ChickSpeak, destinée aux lycéennes américaines. À l'origine, ChickSpeak voulait simplement un réseau de blogs, donc une installation personnalisée de WordPress MU. L'équipe a ensuite demandé de plus en plus de fonctionnalités communautaires à Andy, qui a préféré rester sur la base WPMU+bbPress plutôt que recréer la roue.

Après avoir annoncé cet exploit technique sur son blog, Andy a vite pris conscience de l'engouement de la communauté pour cette orientation de WordPress MU. Il a décidé alors de lancer une version générique et open-source de ChickSpeak, qu'il baptise BuddyPress. Andy a rejoint la société Automattic en 2008 pour travailler à plein-temps sur le logiciel BuddyPress et plus globalement sur l'aspect social de WordPress.com.

BuddyPress fait aujourd'hui partie des projets GPL qu'Automattic a reversés à la WordPress Fundation, et l'équipe de développement a de grands projets – notamment le fait de passer l'intégralité de WordPress.org sous BuddyPress. De fait, son développement est très actif. Pour ce faire, l'équipe de développeurs s'est étoffée, avec l'arrivée de John James Jacoby et de Marshall Sorenson.

Installation dans WordPress

Dans les faits, BuddyPress est un ensemble d'extensions de WordPress et s'installe donc très facilement : il suffit de lancer une recherche sur "buddypress" dans le moteur interne de WordPress, de cliquer sur Installer maintenant et de suivre les instructions. Cela permet de transformer très simplement une installation basique de WordPress en une plate-forme de réseau social.

Après installation des extensions, BuddyPress vous réclamera deux modifications de votre installation :

- mettre à jour vos permaliens ;
- utiliser un thème compatible (BP vous en propose un par défaut).

BuddyPress n'est qu'une surcouche pour WordPress, découpée en plusieurs modules. À l'origine conçu pour ne fonctionner qu'avec WordPress MU (le nom de l'ancienne édition multisite de WordPress), il a depuis été modifié pour fonctionner avec la version autonome de WordPress, même en mode monosite.

Vous retrouvez ainsi huit composants :

- le core (ou cœur du logiciel), contenant les fonctions communes aux différents composants ;
- le flux d'activités ;
- la messagerie privée ;

- les profils étendus ;
- les listes d'amis ;
- les groupes ;
- les forums bbPress ;
- le thème BuddyPress.

Chacun des composants est indépendant des autres, ce qui simplifie le développement et améliore les performances du logiciel. Vous n'installez ainsi que les fonctionnalités qui vous intéressent ! La page Paramètres du menu BuddyPress vous permet d'activer/désactiver six de ces composants. Le thème s'active sur la page des thèmes, et le composant Core ne peut pas être désactivé.

La communauté francophone

Longtemps laissée de côté, l'internationalisation de BuddyPress n'a été réalisée qu'à partir de ses premières versions stables, tandis que la version 1.0 approchait. Celle-ci enfin en place, la traduction a été prise en charge par une nouvelle communauté, BuddyPress France (**http://bp-fr.net/**), menée par Gilbert Cattoire et Daniel Halstenbach, ainsi que Myriam Faulkner.

Cette équipe propose donc une traduction ainsi qu'un site communautaire servant tant de point de rencontre pour les utilisateurs cherchant du support BuddyPress que de site de démonstration de l'outil : en effet, tout comme le site officiel de BuddyPress, le site de BPFR est entièrement conçu avec BuddyPress.

Fonctionnalités

Les profils étendus

Ce module permet à l'administrateur du site de créer les champs contenus dans le profil. Autrement dit, le profil peut contenir tout type d'informations pour mieux répondre aux besoins.

Par exemple, pour le réseau social d'un club de foot, on pourra ajouter un champ comme "poste du joueur sur le terrain". Les formulaires de profil peuvent être classés dans des groupes pour améliorer la lisibilité. Par défaut, BuddyPress crée le groupe "Base", comprenant le champ Name (en réalité, l'identifiant de l'utilisateur sur le réseau). Vous pouvez (et même devez) modifier les noms de ce premier groupe et de ce premier champ dans la page Options générales.

Vous avez tout loisir d'ajouter d'autres informations utiles à votre réseau : prénom, nom, date de naissance, ville. Vous pouvez également créer autant de groupes que nécessaire en plus du groupe Base : Parcours professionnel, Hobbies, Films préférés...

Pour créer ces profils étendus, l'administrateur du site dispose d'un outil puissant, permettant de choisir par exemple le type de l'information (champ texte, liste déroulante, sélecteur de date, etc.).

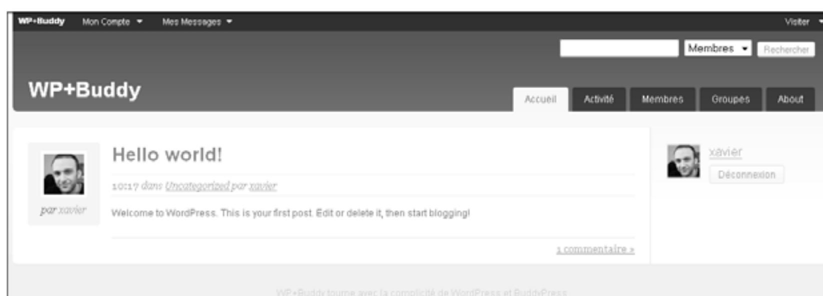
Une fois les champs ajoutés, vous pourrez les réarranger avec un simple glisser/déposer.

Cette fonctionnalité supporte également la gestion des avatars. Chaque utilisateur peut ainsi envoyer une image le représentant. Le service Gravatar est également de la partie : l'avatar par défaut est autogénéré à partir de l'adresse de messagerie de l'utilisateur.

Blog personnel

Autre fonctionnalité pas très originale, BuddyPress permet la création de blogs pour les utilisateurs (voir Figure 16.02). Cette fonctionnalité est fournie directement par WordPress, bien évidemment – si WordPress est en mode multisite.

Figure 16.02
Les blogs intégrés dans BuddyPress.



Les adresses des blogs diffèrent un peu de celles de WordPress. Vous disposez des deux formats suivants :

- *http://monsite.fr/nomdelutilisateur/blog/* ;
- *http://nomdelutilisateur.monsite.fr/blog/*.

BuddyPress rajoute le mot-clef "blog" car il privilégie les profils utilisateur au blog ; vous pouvez cependant modifier le thème pour inverser la situation.

Messagerie privée

Cette fonctionnalité de messagerie e-mail est interne au site (voir Figure 16.03). Les membres peuvent écrire des messages à des contacts de leur liste d'amis, avec bien sûr la possibilité de répondre au message.

Chaque membre dispose de sa propre messagerie : boîte de réception, boîte d'envoi et boîte d'alertes. Le membre est averti pour chaque nouveau message avec un compteur sur l'onglet Messagerie, ainsi que par des alertes e-mail. La messagerie peut utiliser l'éditeur visuel TinyMCE inclus dans WordPress pour faciliter la rédaction des messages.

Figure 16.03
La messagerie privée dans BuddyPress.



La liste d'amis

Les membres du réseau social BuddyPress peuvent se connecter entre eux (voir Figure 16.04). Pour cela, chaque membre peut solliciter la mise en relation avec un autre membre, ce dernier devant accepter la demande pour que la connexion entre les deux soit établie.

Chaque membre possède sa liste publique d'amis. Depuis cette liste, il est possible de naviguer sur les profils des autres membres. Les profils ne subissent aucune restriction. Toutes les informations contenues dans le profil sont visibles par les autres membres. C'est la même chose pour les blogs utilisateur, bien que vous disposiez de la fonctionnalité Article privé ou à mot de passe.

BuddyPress propose la recherche de membres par identifiant, e-mail et nom complet.

Figure 16.04
La gestion des amis dans BuddyPress.



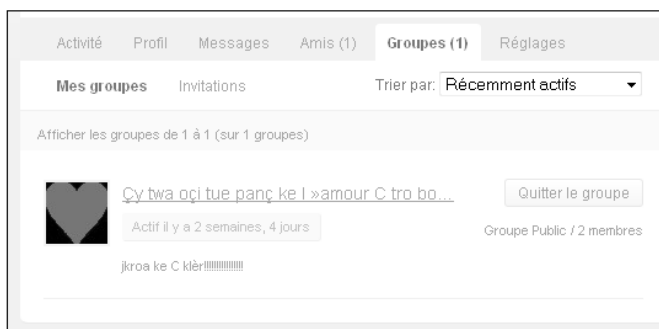
Les groupes

Un groupe dans BuddyPress est un espace de regroupement de membres, d'articles de blogs, de photos et de tout contenu proposé dans WordPress. Tous les membres peuvent créer un groupe dans BuddyPress (voir Figure 16.05). Le membre devient alors l'administrateur de ce groupe, ce qui signifie que c'est lui qui décide, approuve ou rejette les candidatures d'autres utilisateurs voulant rejoindre le groupe.

Mais il ne se résume pas seulement à cela : c'est également lui qui modère le contenu du groupe. Il peut approuver, refuser, et marquer comme spam le contenu proposé par les membres du groupe.

Figure 16.05

Les groupes dans BuddyPress.



Chaque membre de BuddyPress peut écrire des articles sur son blog et dispose de son propre profil. Au lieu de créer un contenu additionnel directement dans les groupes, les membres peuvent taguer leur contenu avec un tag unique (tag généré lors de la création du groupe).

Lorsque le contenu est classé avec le tag du groupe, il est mis dans la liste d'attente de validation de l'administrateur du groupe. Une fois qu'il est accepté, le contenu est visible depuis la page du groupe.

Des liens rapides seront probablement ajoutés pour faciliter l'insertion de contenu dans le blog, et simplifier la vie des utilisateurs.

BuddyPress peut intégrer le forum bbPress pour chaque groupe, afin de permettre des discussions au sein du groupe. Leur installation se fait littéralement en un seul clic.

Les développeurs peuvent facilement étendre les possibilités des groupes ou en ajouter de toutes nouvelles, grâce à une API d'extension.

Le flux d'activités

Cette fonctionnalité donne la possibilité aux amis d'un membre de publier un petit message sur son profil (voir Figure 16.06), qui se présenterait sous la forme d'un texte court et qui pourrait inclure une image et quelques balises HTML de base. Dans les faits, c'est l'un des composants essentiels de tout réseau social, permettant de maintenir un contact entre les utilisateurs. L'éditeur visuel TinyMCE peut être utilisé pour permettre une mise en page rapide.

Figure 16.06
Le flux, clone du mur de Facebook, dans BuddyPress.



Cette fonctionnalité est semblable à celles du thème P2 pour WordPress (**http://p2theme.com/**), autrement dit un usage proche du mur que l'on trouve dans le réseau Facebook. Chaque nouvelle activité dispose de son propre permalien, et les réponses peuvent être affichées de manière hiérarchique afin de simplifier le suivi des conversations entre les commentateurs d'une activité.

Depuis la version 1.1, le flux d'activités comprend également les commentaires des blogs, et ceux des forums.

Mises à jour de statut

Les mises à jour de statut dans BuddyPress apparaîtront sur la page de profil d'un membre. Un membre peut mettre à jour son statut aussi fréquemment qu'il le souhaite. Toutes les mises à jour de statut sont enregistrées dans le journal d'activité du membre.

Les mises à jour de statut s'afficheront dans les résultats de recherche d'ami si les options relatives à la vie privée le permettent.

Le thème de BuddyPress

La première version de BuddyPress nécessitait deux thèmes : un tout à fait standard dans le monde WordPress, affichant la page d'accueil et les blogs du réseau, et le second dédié à l'affichage des pages spécifiques à BuddyPress.

La version 1.1 a corrigé cela : un seul thème peut désormais gérer l'ensemble des pages du réseau social. Le thème du réseau doit se fonder sur un thème parent qui agit comme un framework. Le nouveau thème par défaut utilise ce framework, et peut donc servir d'inspiration.

Il reste bien sûr possible d'utiliser n'importe quel thème WordPress existant et de l'étendre à l'aide du framework afin de fonctionner pour le réseau.

L'avenir

BuddyPress est actuellement en développement intensif. La version a été publiée le 30 janvier 2009, après de nombreux mois d'attente (la première bêta datant du 15 décembre, la première RC, du 11 février). Alors que nous rédigeons ces lignes, la dernière version en date est la 1.2.5.2, qui résout les problèmes de compatibilité entre BP 1.2 et WordPress 3.0...

L'avenir de BuddyPress semble prometteur. Beaucoup de sociétés sont très intéressées par les possibilités du logiciel, aussi bien dans le monde de l'entreprise que dans celui de l'éducation ou encore pour les associations !

17 bbPress – le forum pensé "WordPress"

Présentation

bbPress est un logiciel de forum développé par les créateurs de WordPress. Il fait également partie des projets GPL développés par la société Automattic pour le compte de la WordPress Foundation.

Les auteurs de WordPress ont toujours été frustrés par les logiciels de forum libres existants… Dans leur grande majorité, ces logiciels sont écrits dans un code chaotique et ont une architecture monolithique. Pire, pratiquement aucun d'entre eux ne possède de système d'extensions, ce qui oblige les webmestres à modifier le cœur du logiciel pour ajouter des fonctionnalités. Autrement dit les mises à jour deviennent par la suite un vrai casse-tête.

bbPress se veut un logiciel en rupture avec ce qu'on trouve sur le marché. Là où les alternatives font la course au nombre de fonctionnalités, bbPress est un modèle de simplicité, la qualité du code primant sur la quantité. Pour autant, bbPress n'est pas un logiciel au rabais ; les développeurs ont repris les fonctionnalités qui font le succès de WordPress et ils les ont ajoutées dans bbPress.

Vous retrouvez ainsi un système de taxinomie très puissant, permettant bien plus qu'un classement en forums et sous-forums, une API dédiée aux extensions fonctionnant sur le même principe de hook, des constructions d'URL optimisées pour le référencement, avec un mécanisme de rewriting pensé dès la conception du logiciel, etc.

Enfin, bbPress est livré avec un thème assez particulier (voir Figure 17.01). En effet, contrairement à l'affichage classique des forums, autrement dit une liste de catégories avec à l'intérieur les sujets, bbPress met en avant les derniers messages et les tags du forum. Bien entendu, cette mise en page peut être changée à volonté grâce au mécanisme de thèmes proche de WordPress.

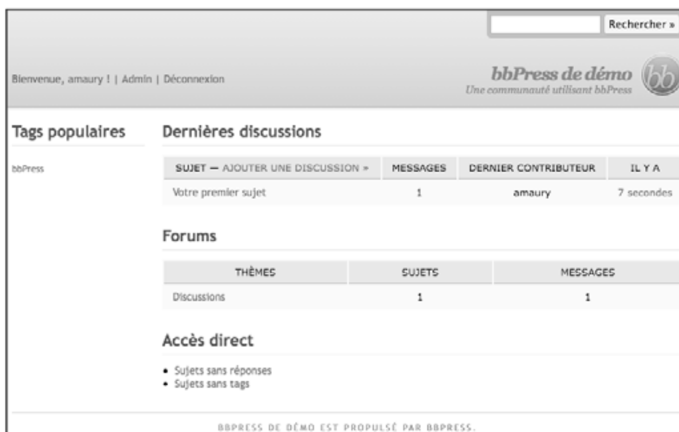


Figure 17.01
bbPress : une interface différente…

Historique

Avant Noël 2004, Matt et son équipe utilisaient le logiciel miniBB pour les forums de support de WordPress.org. Ils avaient vite remarqué les limites de ce logiciel et que le code n'était pas des plus optimisés pour un usage intensif. Chose qui peut paraître aberrante, puisqu'à l'époque miniBB comptait parmi les logiciels de forum les plus légers et les plus rapides, c'est dire…

Après un changement d'hébergement et une mise à jour de PHP, des problèmes étaient survenus sur les forums, et il était devenu alors évident qu'il y avait quelque chose à faire.

Au lendemain de Noël 2004, Matt a débuté le développement d'un nouveau logiciel de forum en partant de rien, et deux jours plus tard la première version de bbPress était mise en production sur WordPress.org.

Depuis, Michael Adams a rejoint l'équipe et a repris les rennes du développement en tant que chef du projet, avec la participation active de Chris Hajer, Sam Bauers et Pete Mall. Ainsi, bbPress continue de croître parallèlement au développement de la communauté WordPress.org.

L'évolution de bbPress restait cependant légère pendant l'ensemble de l'année 2009, ce qu'a admis Matt Mullenweg durant son discours annuel "The State of the Word", où il a promis que le développement de bbPress devrait reprendre de plus belle en 2010, potentiellement sous la forme d'une extension WordPress... Le nouveau design du site officiel est une première indication de cette reprise.

Bloquée depuis de nombreux mois sur la version 1.0.2, l'équipe de développement a annoncé la sortie prochaine de la version 1.1. Celle-ci devrait voir les modifications suivantes :

- abonnement par e-mail ;
- possibilité de publier sans devoir être connecté ;
- meilleure gestion des articles et des sujets ;
- meilleures recherche et statistiques ;
- nombreuses corrections de bugs.

La version 1.1 devrait être sortie quand vous lirez ces lignes...

Fonctionnalités

L'objectif ici n'est pas de lister la totalité des fonctionnalités de bbPress mais plutôt de présenter celles qui apportent une réelle plus-value au logiciel.

Rapide et léger

Le code source de bbPress est pensé et développé de façon à obtenir de très bonnes performances tout en ayant une charge serveur correcte.

Interface simple

L'un des principaux buts de ce projet est de créer quelque chose de simple et d'intuitif à utiliser. Le rêve des développeurs, c'est de faire oublier à l'utilisateur qu'il utilise un logiciel.

D'ailleurs, les utilisateurs de WordPress ne seront pas dépaysés, bbPress se calque sur son interface d'administration (voir Figure 17.02) ; seuls les intitulés, icônes et couleurs des menus changent !



Figure 17.02
bbPress et WordPress : quelques airs de ressemblance.

Thèmes personnalisable

Tout comme WordPress, bbPress propose un mécanisme de modèles pour personnaliser la disposition des éléments. Il est un peu plus compliqué que WordPress, mais sur le concept il utilise la même notion de fonctions PHP dans le code.

Hautement extensible

Là aussi, bbPress reprend un des concepts qui font le succès de WordPress : la possibilité de personnaliser le comportement du logiciel *via* un système d'extensions très puissant.

Protection contre le spam

Akismet, le service permettant de lutter contre le spam, est également disponible pour le logiciel de forum bbPress. Indispensable de nos jours, vu qu'aucun service web n'échappe à ce fléau.

Flux RSS

Vous voulez vous abonner à un sujet ? À un forum ? Aux messages d'un utilisateur ? Dans bbPress, tout est flux RSS !

Par RSS on sous-entend la syndication, plus généralement *via* les formats XML, RSS, RSS2 et Atom.

Intégration facile avec votre blog

WordPress et bbPress sont très proches, ils sont développés par la même équipe, de ce fait l'intégration mutuelle des deux logiciels est très facile.

La taxinomie

bbPress profite des avancées de WordPress pour le mécanisme de taxinomie. Ainsi il est facile de créer sa propre taxinomie et permettre des classements différents par rapport aux classiques forums et sous-forums…

bbPress en français

Depuis 2004, la traduction française de bbPress a régulièrement changé de main. Depuis mars 2008, ce travail de traduction est mené officiellement sous la conduite d'Amaury Balmer.

Un site dédié à bbPress en français, **http://bbpress.fr** (voir Figure 17.03), a été lancé à la même date ; il propose un blog sur l'actualité du logiciel, des traductions, de la documentation et un forum d'entraide.



Figure 17.03
bbPress en français.

Intégration à WordPress

Base d'utilisateurs commune

La première forme d'intégration entre bbPress et WordPress concerne les bases d'utilisateurs. Il est possible de configurer bbPress afin qu'il utilise la base utilisateur, soit la table `wp_users`.

Cette table peut être présente sur la même base de données que WordPress, mais ce n'est pas obligatoire.

D'ailleurs, bbPress peut également se connecter sur une base utilisateur tierce, tout comme WordPress, pour tirer profit par exemple d'une base utilisateur LDAP. Chose que l'on retrouve fréquemment dans les entreprises.

Identification commune

Cette deuxième forme d'intégration a comme prérequis la base d'utilisateurs commune aux deux logiciels. L'identification commune des deux logiciels permet de partager les cookies d'identification.

Cela signifie qu'un utilisateur n'aura besoin de se connecter qu'une seule fois, au forum ou au blog, pour être relié aux deux services. Dans la mise en place d'une communauté, avec un site et un forum, ce type d'identification est obligatoire.

Notez que seules les versions 1.0 et supérieures peuvent se mettre en commun avec WordPress (versions 2.6 et supérieures) ; cela est dû au nouveau mécanisme de génération de cookie qu'intègre la version 2.6.

Correspondance des rôles

bbPress donne la possibilité de choisir la correspondance des rôles avec ceux de WordPress. Grâce à ce système, on pourra par exemple définir quelle est l'équivalence d'un auteur de WordPress dans les rôles de bbPress.

Le logiciel possède six rôles :

- super administrateur ;
- administrateur ;
- modérateur ;
- membre ;
- inactif ;
- bloqué.

Quand intégrer bbPress à WordPress?

Pour intégrer bbPress à WordPress, vous avez deux possibilités. La première se présente lors de l'installation : l'utilitaire d'installation de bbPress propose une étape "intégration à WordPress". Vous avez alors l'occasion de remplir différents formulaires concernant les bases de données utilisateur et l'identification commune.

Vous avez également la possibilité de faire l'intégration dans un second temps ; pour cela, bbPress possède une page Intégration à WordPress dans le menu Réglages.

Cette page donne toutes les indications permettant de réaliser l'intégration des bases, l'identification et la correspondance des rôles entre les deux logiciels.

Partie



Annexes

LE CAMPUS



Participer à l'amélioration de WordPress

Après que vous aurez profité de la grande qualité de WordPress, vous vous sentirez peut-être redevable à ceux qui ont créé ce logiciel et souhaiterez participer à son évolution. C'est là en effet toute la spécificité des projets open-source : tout le monde est invité à participer selon ses capacités, depuis la programmation jusqu'à l'assistance aux débutants et l'amélioration de la documentation.

Vous pouvez améliorer le quotidien des utilisateurs de WordPress de nombreuses manières, et certaines actions ne nécessitent qu'un peu de temps libre et de volonté. Vous pouvez par exemple :

- aider les utilisateurs à résoudre leurs problèmes ;
- traduire WordPress, ses extensions et ses thèmes ;
- améliorer la documentation ;
- participer aux tests fonctionnels ;
- améliorer le code source.

Nous explorerons en détail ces cinq vecteurs d'amélioration dans les pages qui suivent. À la fin de ce chapitre, vous aurez tous les outils en main pour devenir un membre actif de la communauté WordPress.

Aider les utilisateurs

Prérequis : écrire dans un français clair, être patient, savoir se débrouiller, bien connaître WordPress, savoir trouver une information sur le Codex.

C'est peut-être l'une des tâches les plus humbles, mais c'est assurément la plus utile. Tous les jours, des dizaines d'utilisateurs de WordPress se rendent sur les forums d'entraide francophones (**http://www.wordpress-fr.net/support/**, voir Figure A.01) ou anglophones (**http://wordpress.org/support/**) pour exposer leurs problèmes : fonctionnement inattendu de WordPress ou d'une extension, recherche d'informations pour modifier un thème, ajout d'une fonctionnalité, installation impossible, conseils généraux…

Ces forums sont tenus par des bénévoles, dont le temps est limité et qui parfois, simplement, ne savent pas comment répondre à une question. En vous appuyant sur vos connaissances pour répondre aux questions des utilisateurs, non seulement vous aidez une communauté qui repose intégralement sur le bon vouloir de ses membres, mais en plus vous consolidez et étendez votre propre savoir.

Immanquablement, aider les autres à résoudre leurs problèmes se répercutera sur votre maîtrise de WordPress. Tout le monde gagne donc à aider son prochain. Par ailleurs, c'est un investissement qui sera toujours payant : le jour où vous-même aurez besoin d'aide, vous trouverez plus facilement une solution si vous êtes déjà impliqué dans la communauté.



Figure A.01
Les forums du site WordPress Francophone.

Aider des utilisateurs en détresse demande certaines qualités humaines :

- Il faut arriver à cerner le problème qui se pose réellement, car l'utilisateur peut souvent se méprendre sur l'origine de celui-ci.
- Il faut être persévérant, car un problème ne se résout pas toujours du premier coup.
- Il faut avoir de la suite dans les idées, car la solution peut parfois être complexe et nécessiter un vrai travail de recherche.
- Il faut garder la tête froide et rester courtois face à certains utilisateurs très exigeants.

Heureusement, un très grand nombre de problèmes découlent soit d'une méconnaissance de WordPress, soit de la réticence de l'utilisateur à utiliser le moteur de recherche du forum pour trouver la réponse lui-même. De fait, ces problèmes peuvent le plus souvent être résolus avec soit un lien vers la page adéquate du Codex (idéalement, sa version française), soit un lien vers une précédente discussion du forum sur le même sujet – celui-ci existe depuis plus de trois ans, donc le problème a certainement dû être soulevé déjà par un membre (ils sont plus de 8 700, avec environ vingt nouveaux membres par jour) dans une des nombreuses discussions (il y en a plus de 17 000, comptabilisant presque 100 000 messages individuels).

Dans tous les cas, certaines personnes se feront un plaisir d'accompagner vos premiers pas dans l'assistance aux utilisateurs. Parmi eux se trouvent les membres de l'équipe WordPress Francophone (sous les pseudonymes AmO, BenKenobi, Oo, MS-DOS 1991, Damino, ramenian, Xavier et matthieu) et les modérateurs, chargés de s'assurer que les forums sont utiles [Lumière de Lune (Marie-Aude Koiransky), Comme une Image (Jérôme), z720 (Sébastien Erard), FiX (François-Xavier Manet) et claiie (Cyril Clybouw)], sans oublier des utilisateurs particulièrement actifs comme Many (Manalina Rajaona), dlo, bmzoom ou Maître Mô (Jean-Yves Moyart), Rod (Rod Maurice), codfingers (Daniel)… Tous sont bénévoles et désintéressés. N'hésitez pas à leur proposer votre aide, elle sera toujours la bienvenue.

Traduire les textes

Prérequis : écrire en bon français, être bilingue anglais-français, connaître les bases du HTML, PHP et CSS.

WordPress est à l'origine en anglais, mais dans de nombreux pays, des bénévoles prennent en charge la traduction de ce logiciel dans leur langue : corse, arabe, turc, finlandais… WordPress peut être utilisé dans environ 60 langues et dialectes.

La version française est maintenue par l'équipe de l'association WordPress Francophone depuis le début : Xavier Borderie et Amaury Balmer (coauteurs du présent ouvrage) sont les mainteneurs officiels, et font en sorte que chaque nouvelle version de WordPress dispose de sa traduction dans les jours qui suivent la sortie de la version originale – voire le plus souvent dans les heures qui suivent.

Par ailleurs, Amaury Balmer se charge de la traduction de bbPress, ainsi que de l'animation de sa communauté francophone sur le site **http://bbpress.fr/**.

Si la traduction des logiciels WordPress est déjà prise en charge, il est toujours possible de franciser encore plus l'expérience quotidienne du blogueur : comme WordPress, nombreux sont les thèmes et extensions qui sont publiés uniquement en anglais et qui mériteraient une version française en bonne et due forme. Par ailleurs, la traduction de WordPress.com est un processus participatif.

Afin de promouvoir cet aspect participatif de la traduction, les développeurs de WordPress (et en particulier Nikolay Bachiyski, membre bulgare d'Automattic et responsable des aspects linguistiques de WordPress) ont lancé le projet open-source GlotPress, disponible sur **http://glotpress.org/**. Celui-ci est un outil en ligne de traduction collaborative proche de ce que propose le projet Ubuntu avec son outil Launchpad. Officiellement lancé en janvier 2010, GlotPress propulse désormais les sites de traduction officiels des projets d'Automattic, à commencer par WordPress et BuddyPress (tous les deux sur **http://translate.wordpress.org/**), ainsi que WordPress.com (**http://translate.wordpress.com/**). GlotPress peut être utilisé en parallèle au système poEdit+SVN déjà en place...

Traduire un thème ou une extension directement

La traduction directe d'un fichier peut rapidement devenir un processus laborieux : il faut en effet ouvrir tous les fichiers PHP dans un éditeur de texte, comme le Bloc-notes de Windows (et surtout pas Word ou OpenOffice.org), trouver toutes les chaînes de texte en anglais (à ne pas confondre avec les fonctions PHP ou les balises HTML) et les traduire une à une. Ceci fait, le traducteur se charge soit d'envoyer le fruit de son travail au créateur du thème/de l'extension, soit de le diffuser lui-même sur son site ou *via* le site de la communauté WordPress locale (pour la France, WordPress Francophone).

L'aspect laborieux intervient lors de la mise à jour du thème/de l'extension : le processus est alors à revoir depuis le début, car il est quasiment impossible de reprendre soi-même les modifications apportées au programme dans la traduction existante, et copier-coller toutes les chaînes d'un fichier à l'autre peut se révéler tout aussi difficile.

Ce problème récurrent n'est pas unique à WordPress mais concerne tous les logiciels existants, open-source ou non. Pour résoudre ce problème de manière élégante et accessible, le monde de l'Open Source a créé le projet GNU gettext en 1994. Celui-ci a un fonctionnement assez particulier, sans compter une terminologie spécifique, ce qui mérite une explication complète…

Présentation du processus gettext

gettext est une suite d'outils simplifiant grandement la traduction d'un programme. Ils permettent d'extraire les chaînes de texte du code source d'un programme, de les stocker dans un nouveau fichier normé, et de générer un fichier binaire contenant la traduction, fichier qu'un logiciel comme WordPress peut ensuite exploiter pour remplacer automatiquement les chaînes en version originale par leur version traduite.

WordPress supporte le format gettext depuis sa version 1.2 (mai 2004), qui a été traduite en français dans les jours suivants par la communauté.

Le support de gettext nécessite de rendre toutes les chaînes de texte du logiciel (comme WordPress) repérables par l'outil gettext. Ce processus de marquage des chaînes s'appelle l'internationalisation (ou "i18n" pour faire plus court par écrit : un "i", puis 18 lettres, puis un "n") et est obligatoire.

Une fois la totalité du code source du logiciel correctement marqué, il faut lancer sur ce code l'outil gettext, qui va parcourir tous les fichiers, récupérer toutes les chaînes marquées et les rassembler dans un fichier texte utilisant un format propre à gettext, baptisé "fichier POT" (*Portable Object Template*, traduit en "modèle de catalogue"), et ayant pour extension de fichier .pot.

C'est à partir de ce fichier POT que travaillent les traducteurs. Directement à l'aide d'un éditeur de texte, ou plus intelligemment avec un logiciel dédié, ils vont traduire une à une toutes les chaînes du fichier depuis la langue d'origine du logiciel (l'anglais pour WordPress) vers leur langue (le français dans notre cas). Ce processus, qui va toujours de pair avec l'i18n, se nomme localisation, ou "l10n".

Le fichier texte résultant de ce processus se nomme "fichier PO" (*Portable Object*, traduit par "catalogue") et se différencie du fichier POT d'origine par ses chaînes traduites (couplées à leur version originale) bien sûr, mais également par les informations ajoutées par chaque équipe de traduction dans les en-têtes du fichier, concernant notamment la région et le dialecte de la traduction.

Ainsi, l'équipe de WordPress Francophone utilise l'indicateur "fr_FR", le premier pour indiquer le dialecte (le français) et le second pour indiquer la région (la France), ce afin de différencier le travail des équipes utilisant le même dialecte, mais appliqué à des régions différentes, avec leurs usages propres : la traduction fr_FR sera certainement différente de celle de Québec (fr_QC), de Belgique (fr_BE) ou d'autres régions francophones. Idem pour

les autres langues, par exemple pt_PT (portugais du Portugal) diffère de pt_BR (portugais du Brésil), etc.

Voici par exemple le début du fichier fr_FR.po, qui permet de voir les en-têtes du fichier, les fichiers où s'applique chaque chaîne, et le couple chaîne originale/chaîne traduite (respectivement, `msgid` et `msgstr`) :

```
"Project-Id-Version: WordPress 3.0\n"
"Report-Msgid-Bugs-To: wp-polyglots@lists.automattic.com\n"
"POT-Creation-Date: 2010-06-17 16:04+0000\n"
"PO-Revision-Date: 2010-06-17 18:32+0100\n"
"Last-Translator: WordPress Francophone <traduction_at_wordpress-fr.net>\n"
"Language-Team: WordPress Francophone <traduction@wordpress-fr.net>\n"
"MIME-Version: 1.0\n"
"Content-Type: text/plain; charset=UTF-8\n"
"Content-Transfer-Encoding: 8bit\n"
"Plural-Forms: nplurals=2; plural=n>1\n"
"X-Poedit-Language: French\n"
"X-Poedit-Country: France\n"
"X-Poedit-SourceCharset: utf-8\n"

#: wp-admin/admin-ajax.php:36
#, php-format
msgid "<strong>ALERT: You are logged out!</strong> Could not save draft.
<a href=\"%s\" target=\"_blank\">Please log in again.</a>"
msgstr "<strong>ALERTE : vous êtes déconnecté(e) !</strong>
Impossible d&rsquo;enregistrer le brouillon. <a href=\"%s\"
target=\"blank\">Veuillez vous reconnecter.</a>"

#: wp-admin/admin-ajax.php:370
#: wp-admin/edit-link-categories.php:28
#: wp-admin/link-category.php:46
#, php-format
msgid "Can&#8217;t delete the <strong>%s</strong> category: this is the default
one"
msgstr "Impossible de supprimer la catégorie <strong>%s</strong> :
c&rsquo;est celle par défaut"

(...)
```

Une fois le fichier fr_FR.po terminé, il reste à le compiler au format binaire utilisable par le logiciel destinataire. C'est encore un outil gettext qui s'en charge et produit un fichier appelé "fichier MO" (*Machine Object*) : fr_FR.mo. Le fichier MO n'est plus lisible à l'œil nu, mais est particulièrement optimisé pour les logiciels internationalisés.

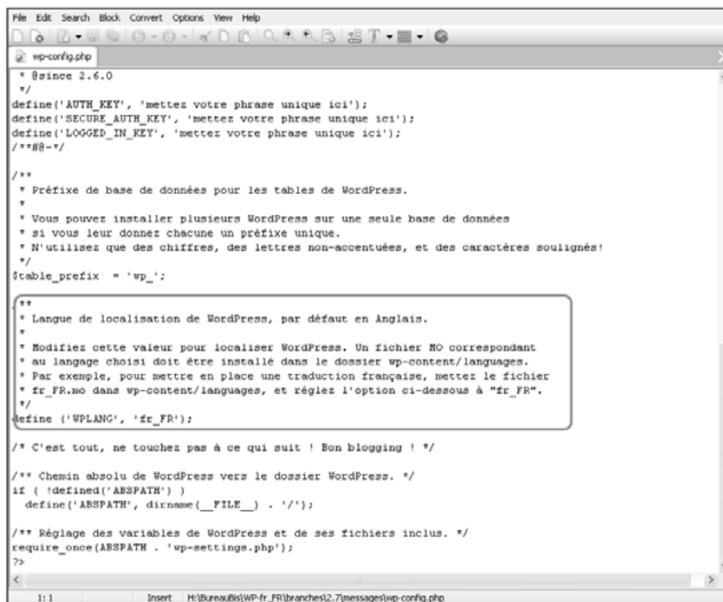
Pour résumer, voici les principales caractéristiques des fichiers POT/PO/MO :

- **POT** (fichier texte). Contient toutes les chaînes non traduites ; généré à partir du code source.
- **PO** (fichier texte). Contient toutes les chaînes dans divers états de traduction ; réglé pour une seule langue ; produit par un outil dédié, comme poEdit.
- **MO** (fichier binaire). Contient toutes les chaînes traduites ; compilé par un outil dédié, comme poEdit.

Il ne reste plus qu'à mettre le fichier MO final en ligne, dans le dossier /wp-content/language, en se conformant à la norme de la langue. Par exemple, pour fr_FR, le fichier devra être nommé fr_FR.mo. Ensuite, il faudra indiquer l'existence de ce fichier à WordPress dans le fichier wp-config.php, à la ligne `define ('WPLANG', '');`, que l'on modifie pour placer le code de langue – pour le français de France, ce sera "fr_FR", donc cette ligne deviendra `define ('WPLANG', 'fr_FR');` (voir Figure A.02).

Figure A.02

Indiquer la langue utilisée dans le fichier wp-config.php.



WordPress se chargera alors de remplacer toutes les chaînes en anglais par leurs traductions françaises en provenance du fichier MO. S'il arrivait que le fichier MO ne dispose pas d'une traduction (par exemple, dans le cas où la version de WordPress serait plus récente que celle pour laquelle le fichier MO a été compilé), gettext afficherait alors la chaîne anglaise originale, au milieu des chaînes traduites.

Jusqu'ici, *a priori*, rien qui semble faire gagner du temps par rapport à une traduction à la main, bien au contraire. Mais c'est lorsqu'une nouvelle version de WordPress sort que le système gettext fait montre de toute sa puissance. En effet, pour chaque nouvelle version de WordPress, majeure ou mineure, un fichier POT propre à cette version est produit, ce qui permet aux traducteurs de suivre l'évolution de cette traduction sans devoir fouiller le code.

Par ailleurs, et c'est là que le génie de gettext se dévoile, les outils gettext sont en mesure d'indiquer aux traducteurs les différences par rapport à la traduction PO précédente : quelles chaînes sont nouvelles, quelles chaînes sont à corriger, et quelles chaînes restent inchangées.

De fait, après le premier travail laborieux de traduction de toutes les chaînes, les mises à jour de la traduction ne sont plus qu'une correction des chaînes modifiées, ou de traduction d'une petite dizaine de nouvelles chaînes au pire. Le travail n'est plus à refaire intégralement, et les traducteurs peuvent se concentrer sur l'amélioration de la traduction plutôt que de reprendre le travail à zéro chaque fois.

Maintenant que vous connaissez la théorie, passons à la pratique…

Localisation d'un thème ou d'une extension

Les projets bien conçus sont déjà internationalisés par leurs auteurs ou la communauté qui les entoure, et ils disposent donc d'un fichier POT à partir duquel vous pourrez travailler. Au pire, leurs chaînes sont déjà marquées selon la convention gettext, et il est dès lors facile de les extraire pour obtenir un fichier POT à partir duquel lancer l'effort de localisation.

Malheureusement, dans la plupart des cas, les thèmes et extensions de WordPress n'ont pas été créés en ayant l'i18n en tête, et il est plus que courant de se trouver avec des archives ne contenant pas de fichier MO de traduction. Dans ce cas, il vous reviendra de contacter son créateur pour le convaincre de l'intérêt de l'internationalisation. Pour ceux qui seraient les plus durs à convaincre, la solution en dernier recours est d'internationaliser vous-même les fichiers et de les fournir au créateur (moyennant la mention de votre nom dans les crédits, bien sûr). Pour savoir comment internationaliser correctement un thème ou une extension – au moyen des méthodes `__()`, `_e()`, `_c()` et des autres –, rendez-vous au Chapitre 10 de ce livre, à la section "Internationalisation de WordPress".

Nous partons du principe que vous disposez du fichier POT du thème ou extension dont vous souhaitez faire une traduction. Vous pouvez utiliser directement les outils gettext, mais ils sont d'un abord assez austère. Il est donc avisé de se servir d'un logiciel dédié. Nous vous conseillons le logiciel open-source poEdit, qui a le triple avantage d'être gratuit, simple à utiliser et multiplate-forme (Windows, Linux et OS X). Il est téléchargeable à l'adresse **http://www.poedit.net/**.

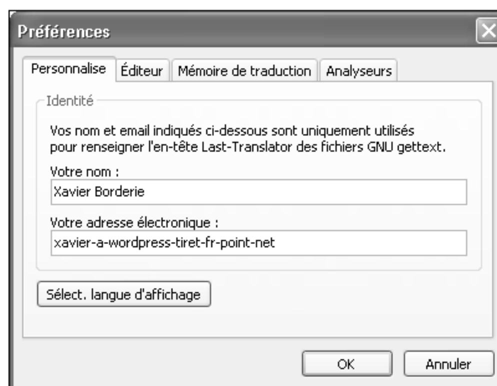
Lors de son premier lancement, poEdit vous demandera dans deux fenêtres consécutives de préciser :

- votre langue préférée, afin de s'afficher dans votre langue (voir Figure A.03) ;
- votre nom et votre adresse e-mail (voir Figure A.04).

Figure A.03
Choix de la langue pour poEdit.



Figure A.04
Saisie du nom et de l'adresse e-mail.



Loin de vous inscrire à une mailing-list, ces deux informations seront utilisées pour "signer" les traductions que vous ferez, et donc vous en attribuer la paternité. Vous pouvez également parcourir les autres onglets pour régler les préférences, par exemple celle qui crée un fichier MO à chaque sauvegarde du fichier PO, mais en général, le nom et l'adresse e-mail suffisent.

Attention, il est possible que la fenêtre où vous pouvez choisir la langue soit cachée derrière poEdit, notamment sur OS X. N'hésitez pas à déplacer la fenêtre principale si vous ne la trouvez pas.

Plutôt que d'explorer les menus de poEdit de manière exhaustive, nous nous contenterons de décrire le processus habituel du traducteur, dans différents contextes…

Commencer une traduction



Si vous préférez mettre à jour une traduction existante plutôt que de démarrer une nouvelle traduction, passez à la section suivante, "Mettre à jour une traduction".

Vous avez le fichier POT fourni par l'auteur du thème/de l'extension, mais aucun fichier PO en vue : vous devez créer ce dernier, qui se différenciera du POT par le fait d'être configuré pour un dialecte précis. Lancez poEdit, et dans le menu Fichier, choisissez l'option Nouveau catalogue depuis un fichier POT (voir Figure A.05). Trouvez le fichier POT dans l'arborescence du thème/extension et chargez-le.

Figure A.05
Création d'un nouveau catalogue.



La fenêtre Configuration s'ouvre immédiatement et vous permet de renseigner les informations liées à votre traduction, et qui feront du fichier POT un fichier PO. Prenez le temps de remplir correctement tous les champs – en vous appuyant sur l'exemple de la Figure A.06, qui présente les informations de la traduction de WordPress.

Figure A.06
Fenêtre Configuration, onglet Info Projet.

L'avant-dernier champ, Jeu de caractère du code source, utilisera généralement "`utf-8`", l'encodage employé par WordPress et recommandé pour tous les projets liés.

Le dernier champ, Formes plurielles, prend une syntaxe particulière, spécifique à chaque langage : la valeur nplural spécifie le nombre de formes, et la valeur plural spécifie à partir de quand la forme est considérée comme plurielle. Pour le français, nous vous conseillons d'utiliser `nplurals=2; plural=n>1` : deux formes (singulier et pluriel), et la forme singulière s'applique à 0 et 1.

Validez ces informations sans vous préoccuper des deux autres onglets. poEdit vous propose d'enregistrer le fichier PO que vous venez de créer.

Voici enfin l'interface principale, divisée en trois sections horizontales (chacune séparée en deux zones) et une barre d'état tout en bas de l'interface (voir Figure A.07).

La section du haut, juste sous la barre de menus, contient toutes les chaînes marquées dans le projet, en provenance du fichier POT : à gauche les phrases originales, à droite les traductions. Les lignes ne contenant pas de traduction sont en gras et en début de liste, tandis que celles avec traduction sont écrites normalement et seront déplacées à la fin de la liste lors de la prochaine sauvegarde.

La section du milieu contient la chaîne à traduire sélectionnée, qui peut comprendre plusieurs phrases, et à sa droite une zone dédiée aux possibles commentaires laissés par les développeurs. Vous ne pouvez pas modifier le contenu de ces deux zones.

La section du bas contient à gauche la zone de traduction. Celle-ci peut proposer deux onglets dans le cas où la phrase originale est susceptible de nécessiter un pluriel – auquel cas il faut traduire la chaîne deux fois : forme singulière dans l'onglet Forme 0, et forme

plurielle dans l'onglet Forme 1. À droite de la zone de traduction, une zone est dédiée aux commentaires que vous pouvez laisser pour vous-même ou pour les traducteurs suivants. Ces commentaires peuvent être des messages informatifs, par exemple des suggestions, des avertissements ou des demandes de vérification en contexte…

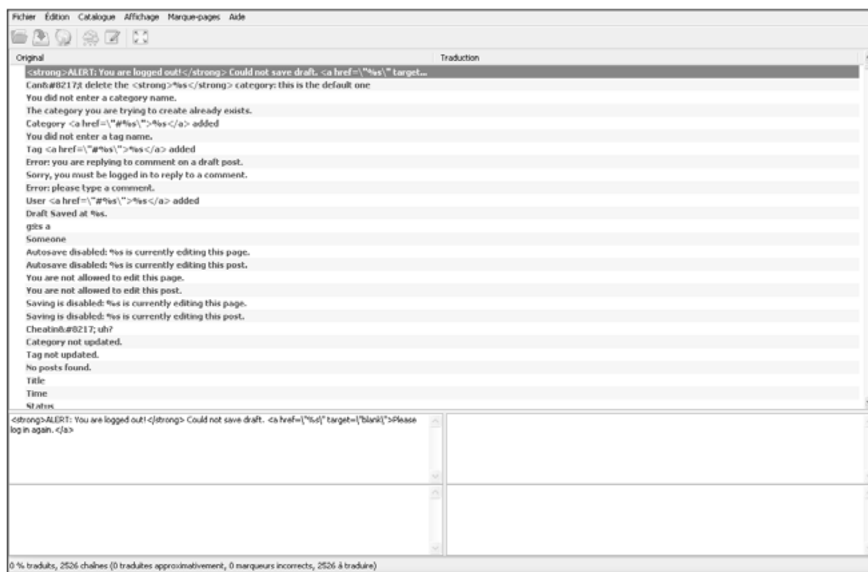


Figure A.07
L'interface principale.

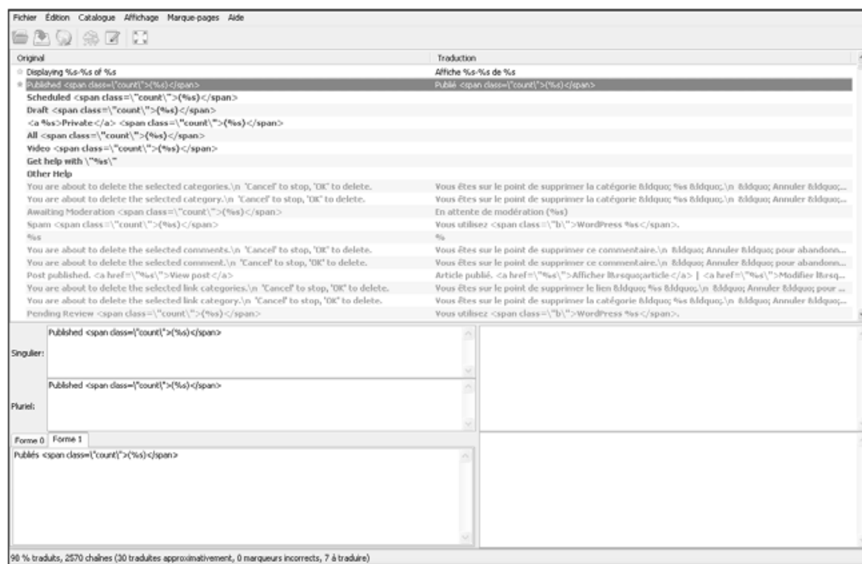


Les deux zones de commentaires seront peut-être invisibles par défaut. Vous pouvez les afficher en passant par le menu Affichage et ses deux options Montrer la fenêtre de commentaires et Montrer la fenêtre automatique de commentaires.

Tout en bas, enfin, la barre d'état vous permet de savoir où vous en êtes de la traduction. D'un coup d'œil, vous connaîtrez ainsi le pourcentage de chaînes traduites, le nombre total de chaînes et entre parenthèses leurs différents types : nombre de chaînes traduites approximativement (fuzzy strings en anglais, c'est-à-dire les chaînes qui ont changé depuis la dernière mise à jour), nombre de chaînes avec marqueurs incorrects (les marqueurs sont des symboles, comme `%s`, `%d` ou `%1$s`, qui permettent au code de placer des variables dans la chaîne sans devoir la couper en deux), et nombre de chaînes restant à traduire (voir Figure A.08).

À partir de là, la tâche est simple : fournir à toutes les chaînes en anglais une traduction en français. Pour accélérer ce travail, il est fortement recommandé de ne pas utiliser la souris pour sélectionner une nouvelle chaîne, mais de tout faire entièrement au clavier : flèche haut/bas pour passer d'une chaîne à l'autre, touche Tab pour sauter directement à la zone où l'on va écrire la traduction, touches Shift+Tab pour revenir à la liste des chaînes. Par ailleurs, en faisant un clic droit sur une ligne, vous obtiendrez tous les fichiers où elle apparaît – ce qui peut être utile pour découvrir son contexte d'utilisation.

Figure A.08
poEdit est prêt à traduire !



Une fois l'intégralité du fichier traduit, récupérez le fichier MO généré automatiquement par poEdit et testez votre traduction – et corrigez-la si nécessaire.

Lorsque c'est fait, il vous reste à faire la promotion de votre dur labeur : envoyez-le à l'auteur ou rendez cette version traduite téléchargeable sur votre site…

Mettre à jour une traduction

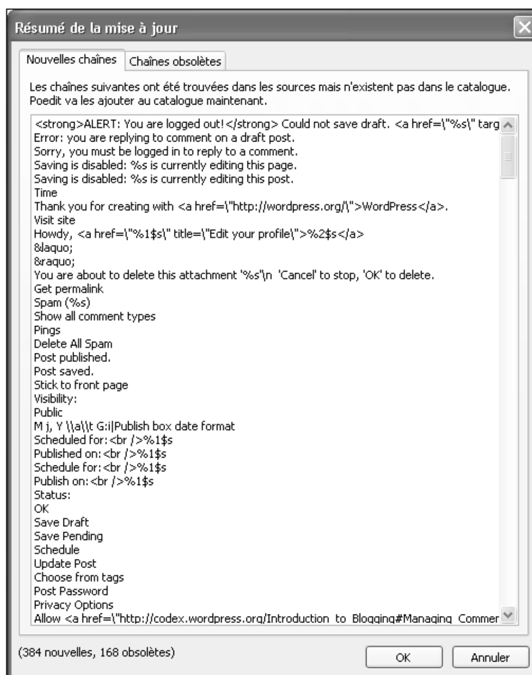
Le fichier PO est entièrement traduit, et le fichier MO turbine désormais joyeusement sur votre blog et peut-être sur d'autres. Félicitations ! Mais voilà que survient l'ennemie jurée des traducteurs : la mise à jour ! En fait, *c'était* l'ennemie jurée, car depuis gettext, le thème "gettexté" (dont les chaînes sont marquées selon le standard gettext) est fourni avec un fichier POT qui contient toutes les chaînes de cette version.

C'est donc ici que gettext démontre tout son talent. Lancez poEdit et ouvrez le fichier PO (celui qui est entièrement traduit) de la version précédente du thème/de l'extension. Ensuite, allez dans le menu Catalogue, cliquez sur l'option Mettre à jour depuis fichier POT et sélectionnez le fichier POT de la nouvelle version. Automatiquement, poEdit va mettre à jour la liste des chaînes et vous indiquera les différences dans une fenêtre de résumé (voir Figure A.09).

Validez, et vous voilà dans l'interface principale, avec les lignes mises à jour : en gras, les chaînes nouvelles (donc à traduire), en rouge les chaînes incertaines (donc à corriger), et en dessous, toutes les chaînes qui ne changent pas entre les deux versions, que vous n'avez donc plus à traduire. Pour certaines extensions simples, ce groupe de chaînes peut être dérisoire, mais quand vous traduisez WordPress et ses presque 2 500 chaînes, vous êtes bien content de n'en avoir qu'une centaine à réécrire…

Figure A.09

Les différences entre deux versions.



Le processus est ensuite classique : traduisez, corrigez, générez le fichier MO, et diffusez-le.

Règles à respecter dans une traduction

Les chaînes à traduire ne seront pas toutes aussi simples que "My taylor is rich". Il faut savoir identifier certains points à négocier proprement.

Les dates

Certaines chaînes ressembleront à ceci : `F jS Y`, `d M Y` ou même simplement `Y`. Dans la vaste majorité des cas, il s'agit là de paramètres envoyés à la fonction `date()` de PHP, pour récupérer la date dans un format particulier. Par exemple, `F jS Y` renverrait une date sous le format "Janvier 3rd 2008", tandis que `d M Y` renverrait "03 Jan 2008". Il vous revient d'adapter le format typiquement anglais, comme notre premier exemple, pour les remplacer par des versions répondant aux coutumes locales. Par exemple, `j F Y` renverra "3 Janvier 2008", ce qui sera beaucoup plus compréhensible.

Tous les paramètres disponibles sont indiqués sur la documentation officielle de PHP : **http://fr.php.net/date**.

Les entités HTML

Vous trouverez également des chaînes avec des codes ésotériques, comme `Newer Entries »` ou `No Comments »`. C'est la solution prônée par les développeurs de WordPress pour mettre en place une sorte de signalétique : les "mots" commençant par &

et se terminant par un point-virgule sont des références alphanumériques à des glyphes (ou caractères) spéciaux. Ces références peuvent prendre deux formes : `»` est une référence prédéfinie pour le glyphe "»", et `»` est une référence numérique pour ce même glyphe. Dans les deux cas, ce sont des codes HTML remplacés par le caractère voulu par le navigateur. Il existe de nombreux autres glyphes, chacun disposant d'une référence numérique, les plus populaires ayant également une référence prédéfinie.

Ces entités doivent être gardées telles quelles, afin de conserver l'esprit donné par l'auteur du script que vous traduisez. Cependant, il est toujours bon de vérifier si certains glyphes ne sont pas trop "locaux" et ne devraient pas être adaptés. Vous en trouverez une liste exhaustive sur Wikipédia : **http://en.wikipedia.org/wiki/List_of_XML_and_HTML_character_entity_references**.

Les espaces

Quand on traduit, il faut connaître certaines règles de ponctuation et savoir les mettre en place correctement. L'un des exemples les plus courants est l'espace. Par exemple, de nombreuses chaînes en anglais contiendront un signe deux-points : "Pages:". C'est ici qu'il faut se souvenir qu'en français un signe de ponctuation peut prendre des espaces avant et/ou après – la règle mnémotechnique étant que si le signe est double (point-virgule, deux-points, point d'interrogation, point d'exclamation), il faut lui adjoindre deux espaces, un de chaque côté. Ainsi, les signes deux-points, point-virgule doivent être entourés d'espaces et non rester collés au mot précédent, comme c'est le cas en anglais. Donc, "Pages:" devrait être traduit par "Pages :".

Seulement le danger est grand de voir ici le deux-points renvoyé à la ligne, par exemple si la mise en page est trop petite. Dans ce cas, il faut bien prévoir d'ajouter une "espace insécable" entre le mot et le signe, en utilisant l'entité " " (nbsp signifiant "non-breaking space"). Cette entité fait que le mot et sa ponctuation ne pourront pas être séparés par un éventuel retour à la ligne automatique. Donc, au final, il faut traduire "Pages:" par "Pages :", et faire de même pour toutes les chaînes contenant un signe de ponctuation double – ou, *in extenso*, pour toutes les chaînes où l'on préfère éviter un retour à la ligne intempestif…

Wikipédia dispose de tous les détails en matière de placement des espaces : **http://fr.wikipedia.org/wiki/Ponctuation**.

Les variables

Les chaînes des thèmes ne sont pas toutes statiques : certaines doivent contenir des informations dynamiques, comme la date, le nombre de commentaires, l'heure – autant d'informations qui ne peuvent pas être écrites "en dur" dans le code HTML du thème.

Sans vous détailler la méthode employée par les développeurs pour placer des variables dans les chaînes – pour les connaisseurs, il s'agit d'utiliser la fonction `printf()` –, voyons le type de chaîne avec variable que vous pourriez croiser : `Published in %s`, `%1$s at %2$s`, voire des combinaisons entité + variable, comme `Archive for the ‘%s’ Category`. Quand une chaîne utilise une variable, elle est marquée `%s` ; quand il y en a plusieurs, le code utilisé est `%x$s`, "x" étant un nombre. Ainsi, s'il y a trois variables dans une chaîne, vous aurez `%1$s`, `%2$s` et `%3$s` – sans limite de nombre.

L'idée ici est de laisser le traducteur placer la variable là où cela aura du sens, en fonction de ses habitudes locales. Cela suppose cependant de deviner à quoi correspond la variable. Par exemple, si le `%s` de `Published in %s` correspond *a priori* à une catégorie (donc, la traduction serait `Publié dans %s`), c'est plus délicat pour `%1$s at %2$s`. Dans ce dernier cas, il faudra faire quelques essais avant de comprendre qu'il peut s'agir d'une variable de date pour `%1$s`, et d'heure pour `%2$s`. La traduction pourrait être `%1$s, à %2$s`, ce afin d'obtenir à l'affichage "lundi 26 janvier 2009, à 12:45", ou même inverser les variables, selon votre préférence, `A %2$s, le %1$s`, ce qui donnerait "A 12:45, le lundi 26 janvier 2009".

Enfin, les combinaisons entité + variable supposent de connaître les entités existantes. Par exemple, `Archive for the ‘%s’ Category` donnerait "Archive for the 'Fruits et légumes' Category". En France, on préférera utiliser les guillemets ouvrants et fermants, dont les codes sont respectivement `«` et `»` ; guillemets que l'on séparera du mot contenu par des espaces insécables au besoin. La traduction donnerait donc : `Archive pour la catégorie « %s »`.

La traduction et la ponctuation française sont un art délicat autant qu'une science épineuse, mais ce n'est qu'en respectant ses contraintes de styles que l'on rendra les chaînes claires pour tous…

Extraire les chaînes marquées pour gettext

Cette étape n'est à accomplir que dans le rare cas où un thème (ou une extension) est déjà internationalisé par son auteur, mais où ce dernier ne fournit pas le fichier POT nécessaire. La tâche vous incombe alors d'extraire vous-même les chaînes du code source et de générer votre propre fichier POT (voir Figure A.10) – qui sera tout aussi valable que s'il avait été donné par l'auteur.



Figure A.10
poEdit est à quelques secondes de générer un fichier POT.

poEdit peut parfaitement s'occuper de cette tâche, en suivant ces étapes :

1. Lancez poEdit pour partir d'une base vierge.
2. Cliquez sur l'option Nouveau catalogue du menu Fichier.
3. Dans la fenêtre qui s'ouvre, entrez le nom du thème/de l'extension et les autres informations que vous jugez nécessaires.
4. Toujours dans cette fenêtre, allez dans l'onglet Chemin et cliquez sur l'icône Créer (deuxième en partant de la gauche, en forme de carré). Une ligne s'ajoute dans la zone en dessous des icônes, dans laquelle vous devez placer le chemin d'accès au dossier du projet. Pour vous simplifier la tâche, mettez simplement un point, et quand viendra le moment d'enregistrer le fichier, positionnez-le à la racine du dossier du projet : de cette manière, poEdit explorera les fichiers du dossier courant.
5. Allez au troisième onglet de la fenêtre, nommé Mots clés. C'est ici que vous indiquerez les marqueurs à chercher dans le code, en l'occurrence `__()` et `_e()` (et au besoin, __ngettext() et `_c()`). Cliquez sur l'icône Créer et saisissez "__" (deux caractères soulignés, ou *underscores)* dans la ligne qui s'affiche. Recommencez la procédure pour ajouter une ligne, dans laquelle vous saisirez "`_e`". Vérifiez bien chaque chaîne : nombreuses sont les fonctions qui touchent à la traduction (voir la section consacrée à ces fonctions), et il ne faudrait pas en rater une…
6. Fermez la fenêtre en cliquant sur le bouton OK. poEdit vous propose d'enregistrer le fichier. Comme nous l'avons conseillé, placez-le à la racine du projet d'où il faut extraire les chaînes. Par défaut, poEdit vous propose l'extension `.po`, mais vous pouvez la changer en `.pot` si vous préférez garder un fichier POT vierge. Validez.
7. Aussitôt le fichier enregistré, poEdit va explorer les fichiers du dossier indiqué et vous afficher les chaînes découvertes dans sa fenêtre Résumé de la mise à jour. Cliquez sur OK, et vous voilà avec un fichier POT qu'il vous reste à traduire pour en faire un véritable fichier PO.
8. Si vous êtes le créateur du thème ou de l'extension, n'oubliez pas d'inclure le fichier POT dans l'archive que vous distribuez. Si vous n'êtes pas le créateur, vous pouvez toujours envoyer le POT à ce dernier pour le pousser à prendre l'internationalisation au sérieux.

Vous avez désormais toutes les clés pour traduire thèmes et extensions à tout-va, et offrir aux utilisateurs francophones une plus grande variété dans leur quotidien de blogueurs.

Utiliser GlotPress pour traduire WordPress.com

WordPress.com est un cas à part : si techniquement la traduction de ce service repose également sur le concept de fichiers PO/MO, ses développeurs ont choisi de rendre sa traduction participative. Ils ont ainsi créé un site à part, **http://translate.wordpress.com**, qui se charge de présenter aux volontaires les chaînes à traduire, dix par dix (voir Figure A.11). Les traducteurs bénévoles peuvent donc s'y connecter (à condition de disposer d'un compte

WordPress.com), traduire autant de chaînes qu'ils le souhaitent en une seule session, et voir leur travail en place sur la plate-forme dès validation par les modérateurs.



Figure A.11
L'interface de traduction participative de GlotPress.

La première page visible du site est **http://translate.wordpress.com/projects**, qui présente les divers projets dont les traductions sont hébergées sur cette installation de GlotPress, projets parmi lesquels se trouve WordPress.com. Un lien Log In en haut à droite vous permet de vous connecter. Ceci fait, vous pourrez choisir le projet auquel vous voulez participer.

En cliquant sur le projet, vous arrivez sur la liste des langues et dialectes disponibles. Cliquez sur la langue vers laquelle vous souhaitez traduire (French (France), ou French (Switzerland), *a priori*), et vous arriverez directement sur l'interface principale de traduction, avec dix chaînes n'attendant que votre participation.

L'interface de GlotPress

Avant de traduire les chaînes à tout va, penchons-nous sur les différents outils qui nous sont offerts. Notez que, selon le statut de votre compte (utilisateur ou modérateur), vous verrez tout ou partie des options de GlotPress...

Le haut de l'interface indique le nom de l'application, suivi du projet et de la langue choisis. Sous ces indications se trouve une suite de liens à gauche, et une pagination à droite. On retrouve cette pagination en bas de page.

Les trois premiers liens (affublés de flèches vers le bas) ouvrent chacun une interface :

- **Bulk.** Lance une action groupée sur les chaînes sélectionnées (au maximum, les dix de la page en cours). Par défaut, un utilisateur ne peut que demander une traduction automatique avec Google Translate. Les modérateurs, eux, peuvent s'en servir pour approuver ou rejeter un lot de traductions proposées.

- **Filter.** Il s'agit en quelque sorte du moteur de recherche de GlotPress. Deux champs servent à indiquer le mot ou l'utilisateur recherché, tandis que les sélecteurs à droite permettent de préciser la recherche : lancer une recherche selon son état de traduction et de validation.
- **Sort.** Moins utilisée que les autres, cette interface sert à trier la liste des chaînes selon certains critères.

Suivent six liens qui agissent directement sur la liste de chaînes affichées :

- **All.** Affiche l'ensemble des chaînes, quel que soit leur état.
- **Untranslated.** N'affiche que les chaînes ne disposant pas d'une traduction.
- **(random).** Affiche une sélection de dix chaînes non traduites et prises au hasard. À noter que dans ce cas la liste n'est pas paginée, ce qui permet sans doute de ne pas se sentir écrasé par le nombre de chaînes restant à traduire...
- **Waiting.** Affiche les chaînes traduites mais pas encore validées par un modérateur.
- **Fuzzy.** Affiche les chaînes dont la traduction doit être revue. Les chaînes traduites avec Google Translate reçoivent automatiquement ce statut.
- **Warnings.** Affiche les chaînes ayant un problème. Le plus souvent, il s'agit de marqueurs qui n'ont pas été bien repris, ou de retours à la ligne surnuméraires...

Traduire les chaînes

Passons à l'interface principale. Typiquement, dix chaînes sont présentées : la version originale à gauche, la traduction (s'il y en a une) à droite. S'il n'y a pas de traduction, vous pourrez lire en grisé "Double clic to add", ce qui signifie "double-cliquez ici pour ajouter [une traduction]". Faites donc cela !

En double-cliquant dans la zone de traduction, un formulaire se présente : c'est ici que vous saisirez et validerez votre traduction. Notez que vous pouvez également ouvrir ce formulaire en cliquant sur le lien Details de chaque ligne.

Sur la section de gauche, vous trouverez la chaîne originale (en gras), et un champ texte vide ou rempli avec la traduction actuelle. Suivent un lien pour recopier la chaîne originale dans la traduction afin de s'épargner un copier-coller laborieux (ce qui arrive souvent, par exemple en cas d'entités PHP ou de liens HTML) et un lien pour demander une traduction automatique *via* Google Translate. Enfin, un bouton permettant de valider votre traduction, ou un lien pour l'annuler.

Sur la section de droite se trouvent les métadonnées de la chaîne : son état actuel, sa priorité, un petit lien "∞" offrant un permalien vers cette chaîne (utile pour discuter d'une chaîne en particulier), ainsi que, le cas échéant, la date d'ajout de la dernière traduction, et l'auteur de cette traduction.

Quand vous aurez validé votre chaîne, GlotPress ouvrira automatiquement le formulaire de la chaîne suivante dans la liste, jusqu'à la dixième. Une fois la dixième chaîne de l'écran validée, il ne vous reste plus qu'à sélectionner un affichage parmi les liens en haut de l'écran

(Untranslated, random, etc.) afin de récupérer dix nouvelles chaînes. Et ainsi de suite, jusqu'à épuisement des chaînes ou de votre temps libre.

Et WordPress.org ?

La version autonome de WordPress dispose également de son GlotPress : **http://translate.wordpress.org**. Ce site héberge de nombreuses traductions collaboratives de WordPress, ainsi que le projet BuddyPress, les extensions GPL de WordPress et les outils mobiles pour Android et BlackBerry.

Le projet WordPress comprend les sous-projets Multisite (les chaînes qui ne sont chargées qu'en mode réseau) et Twenty Ten (le thème par défaut).

En français, WordPress, Multisite et Twenty Ten sont gérés directement par l'équipe de traduction de WordPress-Francophone, sans passer par cet outil. Il est donc inutile de s'en servir à l'heure actuelle. Pour signaler un problème de traduction : **traduction@wordpress-fr.net**.

Améliorer la documentation

Prérequis : avoir un bon niveau en français, bien connaître WordPress, connaître le fonctionnement d'un wiki et du Codex, être bilingue français-anglais.

Bien que la documentation officielle de WordPress soit liée directement depuis l'interface d'administration du logiciel, tous les utilisateurs ne savent pas qu'elle se trouve sur le Codex (**http://codex.wordpress.org** pour l'accueil principal, **http://codex.wordpres.org/fr:Accueil** pour l'accueil en français) [voir Figure A.12], un wiki mis en place par les développeurs de WordPress afin de laisser n'importe qui participer à l'élaboration des documents. Ce wiki utilise le même moteur que Wikipédia et prône la même philosophie : tout le monde est invité à participer.

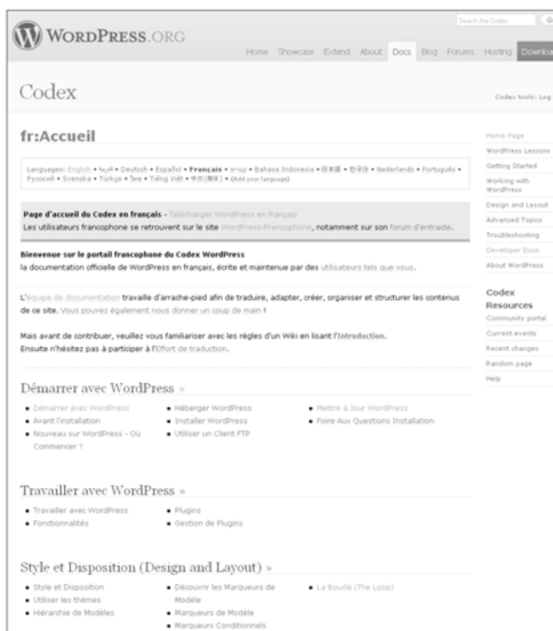


Figure A.12

L'accueil du Codex francophone.

Malheureusement, seule la documentation en anglais est en réalité régulièrement mise à jour, notamment grâce aux efforts de volontaires comme Lorelle van Fossem et Mark Riley (alias Podz). La section française, quant à elle, a de réels soucis pour disposer de pages à jour ou du moins maintenues, parfois de pages tout simplement. Il y a beaucoup moins de volontaires pour cette version française, et les quelques bons samaritains qui s'y lancent ne tiennent jamais longtemps, tant l'effort de rédaction d'un wiki nécessite une saine émulation entre utilisateurs, chacun construisant son apport au tout par des ajouts ou améliorations de ce qui existe déjà.

Écrire une page du Codex reste la façon la plus accessible de participer à l'amélioration de WordPress, pour peu que l'on sache écrire un français correct et que l'on soit capable d'expliquer clairement des concepts techniques.

Participer à l'effort collectif de documentation est aussi le moyen le plus sûr de se rendre utile auprès de la communauté. En effet, la plupart des utilisateurs croient que WordPress est compliqué car ils ne trouvent pas de réponse à leurs questions. Disposer d'une documentation complète et à jour permet donc à tous, utilisateurs comme développeurs, de se reposer sur des textes sûrs et centralisés sur le Codex.

Il y a deux manières de participer à l'effort de documentation : traduire une page en provenance de la version anglaise du Codex, ou créer une page de toutes pièces.

Dans tous les cas, faites savoir que vous souhaitez contribuer à la documentation en vous inscrivant à la mailing-list wpfr-doc (de son nom complet Traduction française de la documentation de WordPress). C'est là que doivent s'échanger les idées entre les différents volontaires. Son adresse : **http://groups.google.com/group/wpfr-doc**.

Cela dit, le mieux pour vous lancer en pratique est de vous inscrire sur le Codex (qui utilise le même moteur que Wikipédia, les habitués ne seront donc pas perdus) et de lire les quelques pages de documentation disponibles depuis la page d'accueil francophone (**http://codex.wordpress.org/fr:Accueil**), notamment les pages Contribuer (**http://codex.wordpress.org/fr:Codex:Contribuer**) et Effort de Traduction (**http://codex.wordpress.org/fr:Effort_de_Traduction**).

Tester WordPress

Prérequis : maîtriser son hébergement web, bien connaître WordPress, être bilingue français-anglais, savoir tracer les origines d'un problème.

Voilà un apport assez particulier au développement de WordPress, mais très précieux. Il est à réserver à ceux qui n'ont pas peur d'installer des versions non stables de WordPress (les versions *bleeding edge*, bêta ou RC), et qui savent comment repérer et surtout signaler les bogues.

Toutes les versions de WordPress, même les mineures, respectent un processus assez suivi : développement pendant plusieurs semaines/mois, période pendant laquelle tout le monde peut récupérer une version bleeding edge (à la pointe), l'installer et voir au jour le jour les

évolutions de WordPress ; mise à disposition d'une première version bêta à tester par les utilisateurs avertis, qui peut être suivie par une deuxième voire une troisième version bêta ; mise à disposition d'une version RC (*Release Candidate*, candidate à publication) pour un test de plus grande ampleur, cette version étant quasiment prête et ne nécessitant que des peaufinages ; enfin, annonce publique de la version finale.

Ces différentes versions de WordPress peuvent être téléchargées par tout le monde :

- **Versions alpha/bleeding edge/trunk.** On les appelle les "nightly" car elles sont archivées tous les soirs à partir des développements du jour écoulé. Le plus simple pour les récupérer est de passer par l'outil de gestion de versions, Trac, hébergé sur **http://trac.wordpress.org/browser** : allez dans le dossier trunk pour avoir accès à tous les fichiers de WordPress et cliquez sur le lien Zip Archive pour les télécharger d'un bloc. Sinon, les habitués de Subversion peuvent directement pointer leur client vers le serveur public de WordPress : **http://svn.automattic.com/wordpress/trunk/**.
- **Versions bêta.** Toujours annoncées sur la liste wp-testers (lire ci-après).
- **Versions RC.** Généralement annoncées sur le blog officiel de WordPress, et avec quelques jours d'avance sur la liste wp-testers.

Les développeurs de WordPress ont énormément besoin de retours de "vrais" utilisateurs pendant les phases de développement, car ils ne peuvent certainement pas tester toutes les conditions possibles.

Si vous souhaitez faire partie de ces valeureux testeurs, inscrivez-vous sur la mailing-list wp-testers, où vous recevrez instructions et conseils, et où vous pourrez lancer des discussions sur les problèmes rencontrés : **http://lists.automattic.com/mailman/listinfo/wp-testers**.

Certains problèmes pourraient être résolus rapidement ou être déjà connus, mais d'autres pourraient justifier la création d'un "ticket", contenant un rapport de bogue en bonne et due forme. La création de tickets Trac est expliquée dans la section suivante.

Améliorer le code source

WordPress, un projet open-source

WordPress est depuis ses débuts un projet open-source, ce qui signifie que son développement est fait selon une méthodologie stipulant entre autres choses que le logiciel doit être librement redistribuable, que son code source doit être ouvert et disponible pour tous, ou encore que n'importe qui peut créer un nouveau logiciel depuis ce code. En pratique, cela signifie que les développeurs qui le souhaitent peuvent participer à l'amélioration du logiciel.

Dans les faits, le développement de WordPress est largement mené par la société Automattic et ses employés, dont le fondateur du projet, Matt Mullenweg. Si tous les développeurs peuvent proposer une modification du code, c'est l'un des membres de l'équipe principale de développement qui décide si elle est intégrée ou non. Outre Matt Mullenweg, les membres de cette équipe sont Ryan Boren (chef de projet), Mark Jaquith, Andrew Ozz et Peter Westwood. WordPress MU a pour chef de projet l'Irlandais Donncha O'Caoimh.

Première approche de Subversion et de Trac

Subversion et Trac sont des logiciels open-source indispensables à tout développeur qui souhaite participer au projet WordPress et à tout développeur en général tant le confort qu'ils apportent est grand.

Subversion est un système de gestion de version (*Version Control System*, ou VCS). Le principe de ce type de système est de stocker sur un serveur (public ou privé) l'ensemble des fichiers d'un projet, avec un historique complet de leurs modifications.

Le développeur travaille sur une copie locale de ces fichiers et ne les envoie sur le serveur que quand il est certain d'avoir terminé son travail. Les autres développeurs peuvent alors synchroniser leur copie locale avec celle du serveur, et les fichiers modifiés sont diffusés ainsi auprès de tous les participants du projet.

L'autre grand intérêt d'un système de gestion de version est qu'en cas de fichier défectueux il est possible de revenir à n'importe quelle version précédente du fichier (ou révision) – à la manière d'un wiki, qui fonctionne sur le même principe. Chaque révision dispose d'un identifiant unique afin de faciliter les retours en arrière, quel que soit le nombre de fichiers auxquels s'applique cette révision. Le projet est donc en sécurité sur le serveur.

Enfin, ce type de système facilite grandement la gestion de *fork*, mot désignant la création d'une version parallèle (ou branche) du projet à partir du code initial, afin d'y développer des fonctionnalités qui reviendront (ou non) dans le projet principal, sans troubler le développement de celui-ci.

Dans Subversion, le projet principal est nommé *trunk* (tronc) ; les projets menés parallèlement sont des branches, et il existe aussi une gestion parallèle nommée *tags*. WordPress utilise le trunk pour tous les développements en cours, les branches pour séparer le code propre à une version majeure donnée (on a ainsi des branches pour les versions 2.0, 2.7 ou 3.0, etc.) ou une version expérimentale (comme la branche "CrazyHorse" pour tester la nouvelle interface graphique de WordPress 2.7 avant de la transposer dans le trunk), et les tags sont assignés aux versions définitives (2.0.11, 2.6.2, 2.7, etc.).

Trac est un système de gestion de projet qui permet d'offrir une vitrine web au développement de projets reposant sur un gestionnaire de version comme Subversion. Trac offre de nombreux outils simplifiant la vie du développeur : gestionnaire de rapport de bogue, historique des modifications, gestion de feuille de route, exploration des versions de fichiers, et même un système wiki pour documenter le projet.

Trac est le compagnon idéal de Subversion : les utilisateurs peuvent y créer un rapport de bogue (un ticket) et peuvent à partir de là dialoguer avec les développeurs, voire soumettre un correctif ; les développeurs peuvent proposer leurs propres correctifs à plusieurs reprises avant de les appliquer au code ; il est possible de voir précisément les différences entre deux versions d'un fichier.

Le projet WordPress utilise Subversion (et son prédécesseur, CVS) depuis ses débuts. N'importe qui peut récupérer le code en provenance du trunk, d'une branche ou d'un tag depuis le serveur public mis en place à l'adresse **http://svn.automattic.com/wordpress/**.

Tous les développements de WordPress se font en public et sont visibles sur le Trac installé à l’adresse **http://trac.wordpress.org/**. Les plus curieux pourront s’intéresser aux dernières modifications et suggestions dans la timeline (**http://trac.wordpress.org/timeline**) [voir Figure A.13], ou voir les tickets des versions à venir sur la feuille de route (**http://trac.wordpress.org/roadmap**) [voir Figure A.14].

Figure A.13

La timeline permet de voir en direct les modifications du code et les bogues trouvés.



Pour les développeurs souhaitant participer, il est indispensable d’utiliser un client Subversion, qui se chargera de synchroniser les fichiers locaux avec les fichiers du serveur.

Sous Windows, nous recommandons fortement le client gratuit TortoiseSVN (**http://tortoisesvn.net/**), qui dispose d’une interface graphique rendant son usage quotidien très aisé.

Sous OS X, il n’y a pas encore de client qui se démarque clairement, mais les choix sûrs sont svnX, SynchroSVN ou SCPlugin, tandis qu’une nouvelle génération arrive avec Versions.app ou ZigVersion.

Nous partons du principe que vous avez déjà installé une version de WordPress à part, en dehors de votre installation principale, pour faire des tests en toute indépendance de thème ou d’extension. En effet, les bogues liés aux thèmes et extensions ne sont surtout pas à rapporter sur le Trac : ce dernier est réservé au code de WordPress, et à lui seul.



Figure A.14
La roadmap, pour savoir à quoi s'attendre pour les prochaines versions.

Soumettre un ticket au Trac de WordPress

Le logiciel Trac centralise donc tous les rapports de bogues de WordPress et les lie aux modifications associées afin de mieux s'y retrouver. Une fois que vous êtes connecté à l'aide de votre identifiant (celui créé pour les forums de WordPress.org), vous n'avez plus qu'à créer un rapport de bogue et le soumettre au Trac *via* un ticket.

Notez que Trac n'est pas le bon endroit pour soumettre des rapports de bogues relatifs à des failles de sécurité, car cela rendrait l'information publique – et donc exploitable par les internautes malintentionnés. Pour ce genre de rapport, envoyez un e-mail à l'adresse **security@wordpress.org**, avec toutes les informations nécessaires pour confirmer la faille.

Créer un rapport de bogue n'est pas une chose à prendre à la légère et nécessite un travail préalable. Avant de vous lancer dans la création du ticket, suivez les indications données ci-après.

Avant même d'écrire le rapport, vérifiez que le bogue que vous avez trouvé n'est pas déjà connu : faites une recherche sur Trac (bouton Search) selon plusieurs critères. Si le bogue est déjà noté dans Trac, vous pouvez décider de laisser les développeurs s'en occuper ou participer à sa correction. Dans ce dernier cas, visitez la page Trac du bogue et lisez toute la discussion afin de voir si votre aide est nécessaire. Si vous avez des détails à ajouter qui pourraient aider sa résolution, alors n'hésitez pas !

Vous êtes arrivé au stade où vous êtes sûr que votre problème n'a pas été repéré. Vous devez maintenant écrire le rapport de bogue. Celui-ci doit être clair et concis : configuration de l'hébergement, description succincte du problème, détails essentiels, et suite d'actions à

accomplir pour reproduire le bogue – un bogue impossible à reproduire est très délicat à corriger. Le but d'un rapport de bogue est de permettre aux développeurs de recréer votre problème sur leur propre installation de test.

Si nous vous conseillons de bien réfléchir avant de créer le rapport de bogue, c'est pour éviter d'inonder les développeurs avec des messages lapidaires et vides d'information, comme "ça ne marche pas, ça marchait hier", "je n'arrive pas à faire ça" et autres remarques…

Par ailleurs, un rapport de bogue ne doit contenir que des certitudes. Il faut impérativement éviter les suppositions et les doutes, comme "je crois que c'est dû à la fonction Untel" ou "à mon avis, vous devriez faire ceci puis cela". Si vous savez comment corriger le problème, il sera toujours temps de proposer un correctif.

Vous voilà fin prêt à écrire votre rapport de bogue avec assurance et maestria. Ouvrez le site Trac à l'adresse **http://trac.wordpress.org/**, connectez-vous (lien Login) et cliquez sur le bouton New Ticket. La page illustrée à la Figure A.15 s'ouvre alors.

Figure A.15
Page de création d'un ticket Trac.

Il s'agit d'un formulaire assez classique, mais où chaque section a son importance et peut influer considérablement sur la rapidité de prise en compte de votre rapport. Vous devez donc bien comprendre les implications de chacune des sections !

- **Summary.** Un résumé en quelques mots du problème, avec déjà un peu de détails : ce résumé sert comme titre du bogue, et les développeurs parcourent tous les jours des centaines de résumés de bogues, soyez donc le plus clair possible – pas de "ça marche pas !" par exemple.

- **Type.** Trac permet de classer les tickets par type ; donc précisez bien si votre rapport concerne un défaut que vous avez repéré (defect), une amélioration que vous suggérez (enhancement), une fonctionnalité que vous voudriez voir apparaître (feature request), ou une tâche générale à accomplir (task, réservée aux développeurs de WP).
- **Description.** C'est le rapport de bogue lui-même. N'hésitez pas à présenter les étapes pour reproduire le problème, à donner les détails de votre hébergement si besoin est, voire à faire un lien vers une page présentant le bogue. Détails, détails, détails…
- **Priority.** Accessible uniquement aux développeurs, il indique une estimation de l'urgence du bogue : très basse (lowest), basse (low), normale (normal), haute (high), et "très haute mondieu onvatousmourir" (highest omg bbq).
- **Component.** Indiquez à quel composant de WordPress s'applique votre ticket : il y en a 61 à l'heure actuelle, et de nouveaux composants peuvent être ajoutés ou enlevés en fonction des besoins du moment. Indiquez bien le composant le plus adéquat à votre rapport, sans quoi il se verrait assigné au mauvais développeur, avec la perte de temps que cela implique.
- **Severity.** Il s'agit ici d'indiquer la portée de votre rapport de bogue et la facilité du correctif qu'il implique : très simple (trivial), facile (minor), normale (normal), demande du travail (major), très important (critical), on ne peut pas sortir de nouvelle version de WordPress tant que ce bogue n'est pas résolu (blocker). Il vaut mieux laisser à "normal", que les développeurs évaluent eux-mêmes l'importance du bogue.
- **Assign to.** Désigne un développeur responsable de la résolution de ce bogue. Si vous savez qui est en charge de la portion de code en question, mettez son pseudonyme. Si vous voulez gérer le code vous-même, mettez le vôtre. Sinon laissez le champ vide.
- **Milestone.** Accessible uniquement aux développeurs, cette section indique la version de WordPress dans laquelle ce bogue devrait être résolu. Cela permet de prévoir le travail à accomplir pour une version – mais, si le bogue n'est pas corrigé, les développeurs peuvent choisir de le reporter à une version suivante.
- **Version.** La version de WordPress dans laquelle vous avez découvert le bogue. Idéalement, vous avez testé la toute dernière version stable de WordPress, ou la version en cours de développement (trunk) ; si c'est le cas, mettez le numéro de version dans ce champ ; sinon laissez-le vide.
- **Keywords.** Liste de mots-clefs permettant d'identifier rapidement le bogue. Les mots-clefs doivent être séparés par une virgule. Évitez d'en mettre plus de deux, et restez concis. Par exemple, "posting" indique que le bogue s'applique aux fonctions relatives à la mise en ligne. Les développeurs sont susceptibles d'ajouter d'autres mots-clefs, ce qui leur permet de mieux trier les tickets : reporter-feedback (besoin de plus d'informations), needs-patch (ce bogue a besoin d'un correctif), has-patch (un correctif est prêt), needs-testing (le correctif doit être vérifié), 2nd-opinion (besoin d'un avis supplémentaire sur le bogue ou le correctif), tested (un des correctifs a été testé), etc. De fait, si vous avez un correctif à proposer pour un ticket, n'oubliez pas d'ajouter le mot-clef has-patch, vous attirerez ainsi l'attention d'un développeur…

- **Cc.** Désigne des développeurs qui ne sont pas responsables du bogue, mais qui voudraient sans doute être tenus au courant de son évolution. Si vous ne souhaitez pas mettre votre pseudonyme dans le champ Assign, indiquez-le dans ce champ et vous recevrez un e-mail à chaque ajout fait au ticket.
- **I have files to attach to this ticket.** Cochez cette case si vous souhaitez mettre en ligne un fichier qui aidera à la résolution du bogue que vous décrivez : capture d'écran, correctif…

Tous les champs sont maintenant remplis ; validez le formulaire. Si vous avez coché la case en fin de formulaire, Trac vous proposera de mettre en ligne un fichier. Une fois le ticket posté, vous pouvez vous féliciter d'avoir participé à l'amélioration de WordPress !

Trouver des bogues à corriger

Découvrir un nouveau bogue reste une activité aléatoire, et vous souhaitez peut-être vous plonger immédiatement dans la programmation plutôt que d'attendre d'avoir trouvé votre bogue à vous. C'est possible !



Figure A.16
La liste des comptes rendus.

En effet, tous les tickets répertoriés dans Trac n'ont pas forcément de correctif ou de développeur assigné, et attendent qu'on s'occupe d'eux. Ces tickets sont clairement regroupés par Trac dans ce qu'on appelle des comptes rendus, accessibles depuis le bouton View Tickets (voir Figure A.16). Ces comptes rendus trient les tickets en fonction de leur état, de leurs mots-clefs ou d'autres données. Vous pouvez donc d'un clic afficher le compte rendu des tickets actifs, assignés, bloquants, en mal de documentation…

Figure A.17
Le compte rendu pour les tickets en manque de correctifs.

Les comptes rendus qui vous intéressent seront principalement ceux qui listent les tickets en manque de correctifs (Needs Patch, nº 13) [voir Figure A.17] et ceux qui n'ont pas de fichier joint (Lacks Attachment, nº 18). De manière générale, vous êtes libre de vous intéresser à tous les tickets – il est possible qu'un correctif existant ne soit pas optimal, par exemple.

Idéalement, vous choisirez un ticket où vous savez quoi faire pour corriger le bogue décrit ; ne partez pas au hasard, il est préférable de travailler avec un objectif clair en vue. Vous devez ainsi maîtriser le langage de programmation nécessaire (HTML, PHP, CSS ou JavaScript) et le composant en cause (API de thème, API d'extension, API de commentaire, TinyMCE, jQuery, etc.).

Il vous reste maintenant à créer le correctif pour le bogue sur lequel vous avez jeté votre dévolu.

Créer un correctif au format .patch ou .diff

Vous savez comment corriger un bogue déjà listé dans Trac (par vous ou un autre) : formidable ! Mais il est hors de question d'expliquer comment le corriger dans le ticket Trac, non seulement parce que certains bogues demandent des correctifs sur plusieurs fichiers, mais aussi parce qu'il existe une méthode optimale pour agir : fournir le correctif directement en fichier joint du ticket, correctif que les développeurs n'auront plus ensuite qu'à tester et appliquer. Cette méthode implique que vous ayez installé un client Subversion sur votre

machine de développement – c'est important, car c'est Subversion qui va récupérer le code de WordPress sur lequel vous allez travailler.

Décidez quelle version de WordPress vous allez utiliser pour corriger le bogue, en faisant en sorte qu'elle corresponde bien à la version à prendre en compte : le trunk si vous travaillez sur la version en cours de développement, la bonne branche ou le bon tag si vous travaillez sur une version spécifique et déjà diffusée. Pour notre exemple, nous allons partir du principe que vous travaillez sur la version en cours de développement ; si ce n'est pas le cas, vous devrez adapter ce qui suit…

Vous disposez de Subversion et savez sur quelle version vous allez travailler. Vous devez maintenant dire à l'un de télécharger l'autre. Chaque client Subversion a son fonctionnement propre, nous nous baserons sur celui de TortoiseSVN, meilleur client SVN à ce jour, et malheureusement limité à Windows :

1. Créez un nouveau dossier où seront téléchargés les fichiers de WordPress. C'est dans ce dossier que se feront tous vos développements WordPress. Trouvez-lui donc un emplacement pérenne.
2. Lancez une récupération SVN sur ce dossier. Avec TortoiseSVN, faites un clic droit sur le dossier et choisissez l'option SVN Checkout dans le menu contextuel.
3. Indiquez l'adresse du dépôt Subversion (repository en anglais) à partir duquel télécharger les fichiers :
   - Si vous travaillez sur le trunk, ce sera **http://svn.automattic.com/wordpress/trunk/**.
   - Si c'est une branche de code, par exemple 3.0, ce sera **http://svn.automattic.com/wordpress/branches/3.0/**.
   - Si vous travaillez sur une version spécifique, par exemple la 3.0.1, ce sera **http://svn.automattic.com/wordpress/tags/3.0.1/**.

   Assurez-vous que vous prenez bien la dernière révision du code (HEAD revision en anglais).
4. Le client Subversion récupère les fichiers. Vous pouvez voir l'évolution du téléchargement *via* la fenêtre qui s'affiche. Une fois le téléchargement terminé (la fenêtre ne défile plus et affiche Completed avec la mention At revision XXXX), fermez la fenêtre en cliquant sur OK.
5. Installez WordPress localement, afin de lancer des tests. Vérifiez que vous pouvez bien utiliser WordPress localement, et surtout que vous pouvez exactement reproduire le bogue du ticket visé.
6. Corrigez le bogue directement dans le code de WordPress, et pas dans un autre dossier. Vérifiez localement que vous n'arrivez plus à reproduire le bogue.

TortoiseSVN modifie les icônes des fichiers et dossiers : s'ils n'ont pas été corrigés, ils ont une coche verte ; s'ils ont été rectifiés, ils ont un point d'exclamation rouge. Cela permet de mieux s'y retrouver.

Si vos modifications vous déplaisent et si vous voulez revenir au code original, lancez un retour en arrière (revert en anglais) : avec TortoiseSVN, sélectionnez les fichiers à remettre en état, faites un clic droit, et dans le menu contextuel, choisissez l'option Revert du sous-

menu TortoiseSVN. Une fenêtre apparaît, vous demandant de valider les fichiers à remettre en état. Validez. Vous avez à nouveau les fichiers originaux – toutes vos modifications ont été perdues.

Si votre correctif nécessite la création d'un nouveau fichier (ce qui est à déconseiller), n'oubliez pas de l'ajouter dans la base de données de Subversion. Avec Subversion, il suffit de faire un clic droit sur le fichier créé et de choisir l'option Add… du sous-menu de TortoiseSVN.

1. Exportez le code modifié dans un fichier au format .diff. Ce format de fichier texte, propre à Subversion, indique uniquement les lignes qui ont changé, et par quoi elles ont été remplacées – donc, les différences par rapport à la version originale du code. Pour exporter le .diff avec TortoiseSVN, faites un clic droit sur le dossier principal et choisissez l'option Create Patch du sous-menu TortoiseSVN. Une fenêtre vous propose de choisir les fichiers à partir desquels tirer les différences – ce qui est nécessaire, car vous pouvez avoir plusieurs correctifs en cours qui ne s'appliquent pas tous au même ticket. Validez et enregistrez le fichier sous un nom compréhensible. Ensuite, TortoiseSVN vous affiche le contenu du fichier.
2. Joignez votre fichier correctif au ticket Trac approprié. Ouvrez la page du ticket dans votre navigateur et cliquez sur le bouton Attach File, situé sous le descriptif du bogue. Un formulaire vous est alors présenté : trouvez votre correctif *via* l'explorateur de fichier, ajoutez une courte description si besoin est, cochez la case si votre correctif doit remplacer le même fichier du même nom déjà présent sur ce ticket, et validez en cliquant sur le bouton Add Attachement (voir Figure A.18).
3. Pour parfaire la chose, une fois le correctif correctement inséré, ajoutez un commentaire au ticket pour décrire votre travail, puis saisissez le mot-clef "has-patch" dans le champ de description Keywords.
4. C'est terminé ! Félicitations, et merci pour votre contribution à l'amélioration de WordPress !

Figure A.18
Formulaire d'ajout de correctif à Trac.

Grâce à Subversion, si votre correctif est validé par les développeurs, ils pourront directement l'appliquer sans même toucher au code : c'est Subversion qui se chargera de placer vos lignes de code corrigées aux bons endroits, automatiquement – c'est là tout l'intérêt d'utiliser un gestionnaire de version.

Les fichiers .diff utilisent un format textuel normé et propre à Subversion. Pour autant, ces fichiers sont parfaitement lisibles. Par exemple, mettons que votre correction consiste à modifier le fichier `readme.html` pour changer "Matt Mullenweg" en un autre nom. Voici ce que contiendrait le fichier `.patch` généré par TortoiseSVN :

```
Index: readme.html
===================================================================
--- readme.html  (revision 9125)
+++ readme.html  (working copy)
@@ -14,7 +14,7 @@

 <h1>First Things First</h1>
 <p>Welcome. WordPress is a very special project to me. (...)
-<p style="text-align: right;">&#8212; Matt Mullenweg</p>
+<p style="text-align: right;">&#8212; Xavier Borderie</p>

 <h1>Installation: Famous 5-minute install</h1>
 <ol>
```

Vous pouvez voir que le nom du fichier est clairement indiqué, ainsi que la ligne qui a été modifiée : l'originale est précédée d'un "-" pour indiquer qu'elle va être supprimée, et votre version de cette ligne est précédée d'un "+" pour indiquer qu'elle remplacera dans le fichier la ligne qui vient d'être effacée. Par ailleurs, le symbole `@@` indique la ligne où commence le texte affiché : en effet, le correctif contient également quelques lignes non rectifiées autour de la ligne modifiée (généralement trois avant et trois après), afin de conserver un contexte et de rendre le fichier plus lisible.

Notez que si vous utilisez la ligne de commande "`svn`", vous pouvez directement lancer la récupération avec la commande (une fois que vous vous trouvez dans le dossier cible) `svn co http://svn.automattic.com/wordpress/trunk/` et créer le fichier .diff avec `svn diff > my-patch.diff`. Attention, cette commande crée un fichier .diff de toutes les modifications, sur tous les fichiers. Si vous voulez vous limiter à un seul fichier, utilisez la commande `svn diff dossier/nomdufichier.php > my-patch.diff`. Pensez également à donner un nom significatif au fichier, idéalement en ajoutant le numéro de la révision à laquelle il s'applique.

Cette annexe se conclut ici. Vous avez désormais en main toutes les clés pour devenir un membre actif de la communauté WordPress. Lancez-vous sans hésiter !

B Description du schéma de la base de données MySQL de WordPress

Concept

WordPress utilise depuis sa création la base de données MySQL. Cette base a l'avantage d'être libre, très répandue chez les hébergements web et de bonne facture.

Avant WordPress 2.1, le moteur était compatible avec la version 3 de MySQL. L'intérêt de ce prérequis très faible était la possibilité d'installer WordPress sur tout type de serveur, même ancien.

Néanmoins, afin de répondre aux besoins des développeurs, le passage à la version 4 de MySQL a été obligatoire, car cette version apporte notamment le support des sous-requêtes, bien plus pratiques et plus performantes que l'exécution de deux requêtes SQL distinctes. Depuis WordPress 2.1, les prérequis liés à MySQL n'ont pas changé.

Avec WordPress 2.3, les développeurs de WordPress ont largement remanié le schéma de la base de données. En règle générale, la philosophie des développeurs est d'apporter des modifications mineures au schéma, permettant ainsi de garder une compatibilité avec les extensions existantes. Pour la version 2.3 et l'apport de la nouvelle API de taxinomie, les développeurs ont dû supprimer trois tables complètement spécifiques pour en créer trois génériques.

Ces trois tables permettaient de catégoriser les liens et les articles, mais chacune de leur côté. Avant WordPress 2.3, les tags n'étaient même pas gérés en natif !

Désormais, l'API de taxinomie permet de gérer autant de classification que l'on souhaite, mais sur trois tables MySQL ! (Voir les explications dans la section "La taxinomie dans WordPress" au Chapitre 10.)

Avec WordPress 3.0 et l'intégration du mode multisite, le schéma de base de données évolue en fonction du mode d'utilisation de WordPress, comme c'est expliqué en fin de chapitre.

Pour le reste, nous allons voir l'usage de chaque table.

Les tables en mode monosite

wp_commentmeta

Cette table stocke toutes les métadonnées liées aux commentaires. C'est ici que les extensions ajouteront leurs propres informations.

wp_comments

Cette table stocke les commentaires et rétroliens des articles et pages. Elle est rarement utilisée par les extensions, car la structure de la table ne s'y prête guère. Néanmoins, il est possible d'ajouter des données en plus des commentaires et rétroliens grâce à un champ `comment_type` qui définit le type du contenu.

wp_links

Cette table stocke les liens de la blogoliste. Cette table est peu exploitée par les extensions, car on touche une fonctionnalité secondaire de WordPress

wp_options

Cette table contient l'intégralité de la configuration de votre blog, elle est utilisée par l'API des options. On peut y stocker de façon très simple la configuration de son extension ou des valeurs nécessitant peu d'enregistrement en base de données.

wp_postmeta

Cette table contient les métadonnées des articles, on y retrouve des informations annexes aux articles et pages, par exemple le template utilisé par une page, etc. Cette table est générique, ainsi elle peut être utilisée pour stocker tout type d'informations en relation avec un article ou une page. Par exemple, on peut y enregistrer la notation d'un article, le compteur de vue, etc. Beaucoup d'extensions travaillent avec cette table.

wp_postmeta

Chaque article dispose d'informations liées, nommées métadonnées. Elles sont stockées dans cette table. Les extensions s'en servent fréquemment.

wp_posts

Cette table stocke les articles et les pages. Techniquement, on y retrouve aussi les révisions des articles et les pièces jointes à un article (par exemple les images envoyées *via* le gestionnaire de médias de WordPress), ainsi que les menus de navigation. Cette table peut également être utilisée par les extensions pour stocker d'autres types de données : en effet, chaque contenu dans la base possède comme information le `post_type` qui revient à préciser si c'est une page, un article ou autre chose.

wp_terms

Cette table permet de stocker les termes dans WordPress. Un terme n'est pas nécessairement une catégorie ou un tag, il peut s'agir des deux en même temps (voir la section "La taxinomie dans WordPress" au Chapitre 10 pour comprendre la subtilité).

wp_term_relationships

Cette table permet de faire la relation entre les contenus (articles, liens, pages) et un élément de taxinomie (catégorie, tags, etc.).

wp_term_taxonomy

Cette table permet de créer un contexte pour chaque terme. Par exemple, on pourra retrouver le contexte catégorie et le contexte tag pour un même terme.

wp_usermeta

Cette table contient les informations annexes à un utilisateur, soit les différentes informations présentes sur la page de profil (messagerie, description) ainsi que des données d'administration comme le rôle ou le niveau de l'utilisateur.

wp_users

Cette table contient les données principales sur les utilisateurs de WordPress, dont l'identifiant, le mot de passe, le nom à afficher, l'adresse e-mail, etc.

Les tables en mode multisite

En plus des tables ci-dessus, le mode multisite ajoute ses propres tables à la base de données.

À noter qu'en plus de la création de ces tables le mode multisite ajoute quelques champs aux tables `wp_usermeta` et `wp_users`.

Enfin, chaque site dispose de son propre jeu de tables WordPress, sous la forme `préfixeDeTable_idDuSite_nomDeLaTable`, par exemple avec "wp" pour préfixe et le quarante-deuxième site du réseau, la table contenant les articles serait wp_42_posts.

wp_blogs

Chaque site créé est stocké dans cette table.

wp_blog_versions

Cette table maintient une liste de l'état de la base de données de chaque site. Elle n'est modifiée que lorsque le site est mis à jour.

wp_registration_log

Cette table contient le nom de l'utilisateur/administrateur de site chaque fois qu'un site est créé.

wp_signups

Cette table garde un journal de tous les utilisateurs ayant créé un site.

wp_site

Cette table contient l'adresse du site principal.

wp_sitecatagories

Dans le cas où les termes globaux sont activés pour un site, cette table contiendra ces termes.

wp_sitemeta

Chaque site dispose d'informations qui lui sont liées, comme le nom de son administrateur. C'est cette table qui stocke ces données.



Index Utilisateur



Index Développeur

LE CAMPUS

L

M

N

O

P

R

S

Le Campus

WordPress 3

Vous souhaitez créer, personnaliser et gérer facilement un blog, un site web professionnel ou même un réseau de sites ? WordPress est le *nec plus ultra* des plates-formes de publication personnelle, alliant esthétique, standards du Web et utilisabilité. Système de gestion de contenus (CMS) gratuit et open-source, il est l'un des outils de blog les plus populaires et fait tourner plusieurs millions de sites web.

Ce livre, écrit par trois personnalités de la communauté francophone et internationale de WordPress, est un véritable guide de référence. Il vous fera entrer de plain-pied dans la communauté des utilisateurs aguerris en vous livrant l'ensemble des informations utiles pour comprendre le logiciel et en exploiter tout le potentiel.

Cette seconde édition, entièrement révisée, couvre toutes les nouveautés de la version 3. Les débutants pourront construire un blog riche et personnel, et les lecteurs plus expérimentés approfondiront leurs connaissances techniques pour concevoir thèmes et extensions, mais aussi mettre en place un réseau de sites à l'aide de la fonctionnalité multisite intégrée. Vous détiendrez alors toutes les clés pour prendre entièrement en main votre site.

À propos des auteurs

Xavier Borderie est mainteneur officiel de la traduction française de WordPress depuis 2004, co-fondateur de la communauté d'entraide WordPress Francophone lancée en 2005, et président de l'association du même nom.

Francis Chouquet est webdesigner et créateur de thèmes pour WordPress, ainsi que consultant web spécialisé dans les blogs.

Amaury Balmer est co-fondateur et responsable technique du site WordPress Francophone, et l'un des mainteneurs officiels de la traduction de WordPress, ainsi que de celle de bbPress. Il a également créé plusieurs thèmes et extensions de WordPress, dont Simple Tags.

Fichiers d'exemples sur http://wordpress-apprendre.fr/livre

Table des matières

Niveau : Débutant/intermédiaire
Catégorie : Web
Configuration : Multiplate-forme

PEARSON

Pearson Education France
47 bis, rue des Vinaigriers
75010 Paris
Tél. : 01 72 74 90 00
Fax : 01 42 05 22 17
www.pearson.fr

ISBN : 978-2-7440-4162-4

